Einführung in Visual Computing

186.822

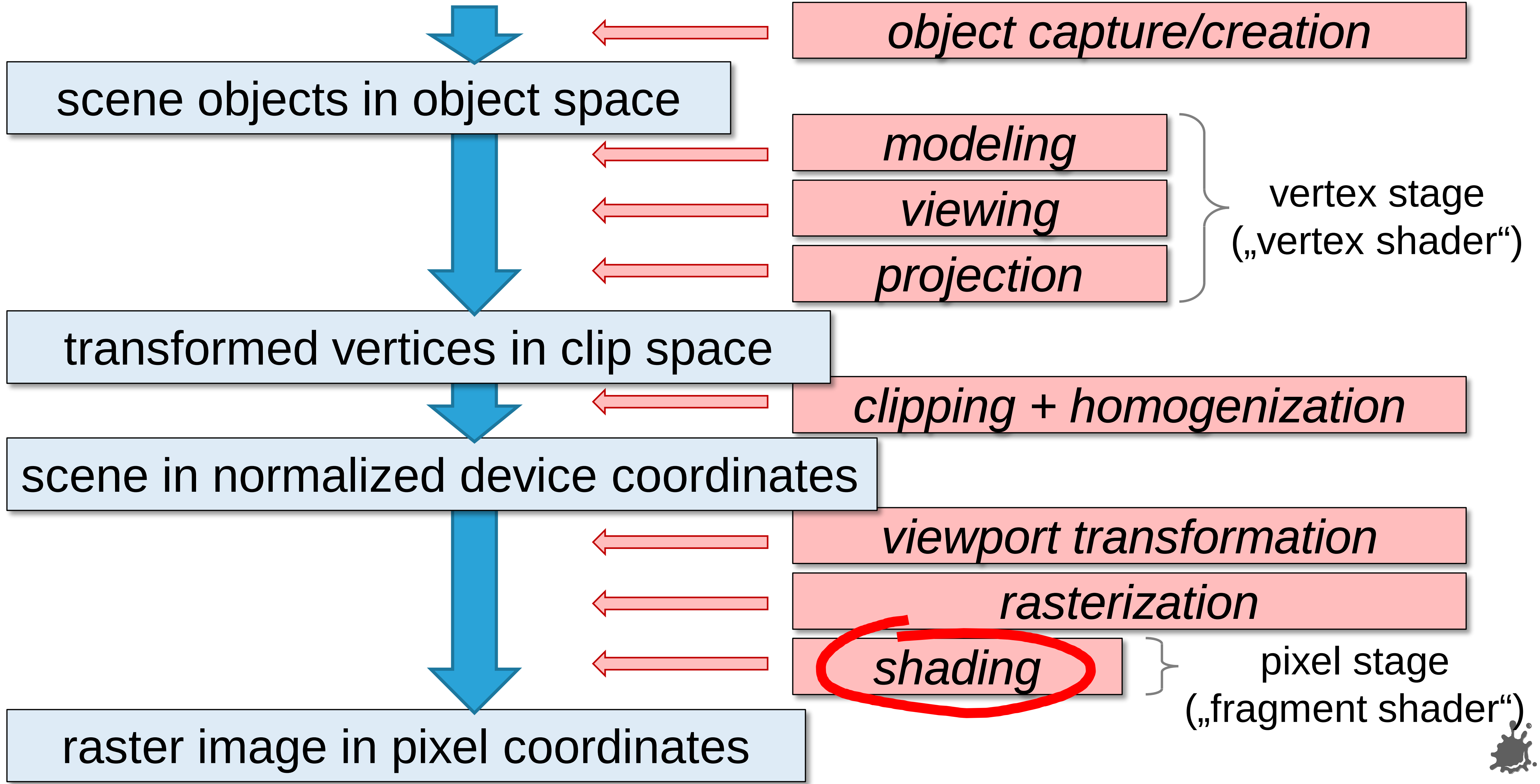
Textures

Werner Purgathofer

Surface-Rendering Methods

- polygon rendering methods
- ray tracing
- global illumination
- environment mapping
- texture mapping
- bump mapping

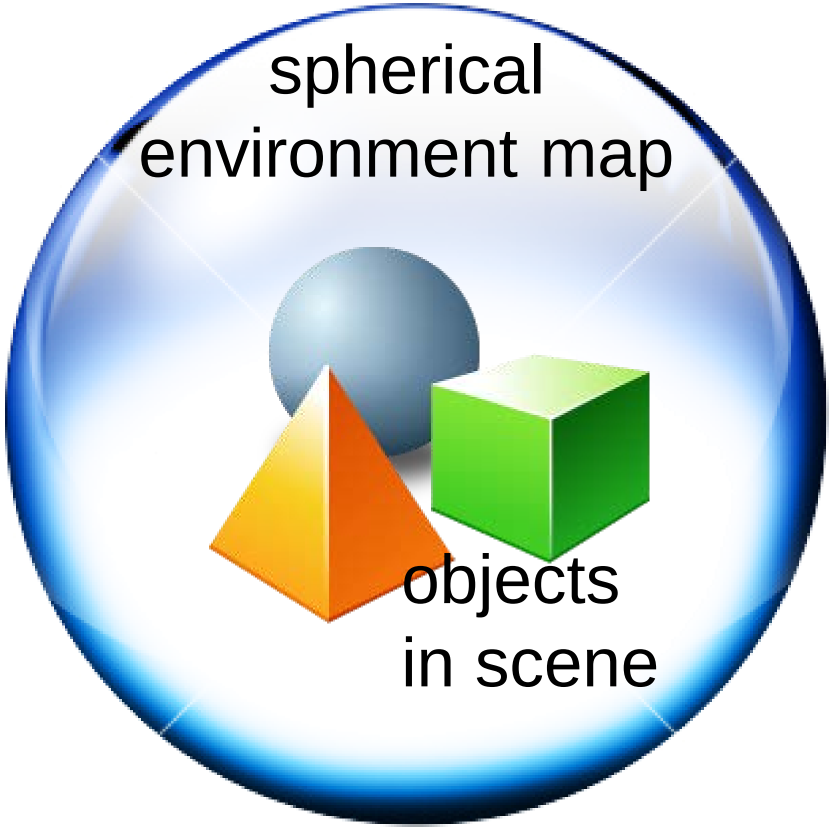
Textures in the Rendering Pipeline

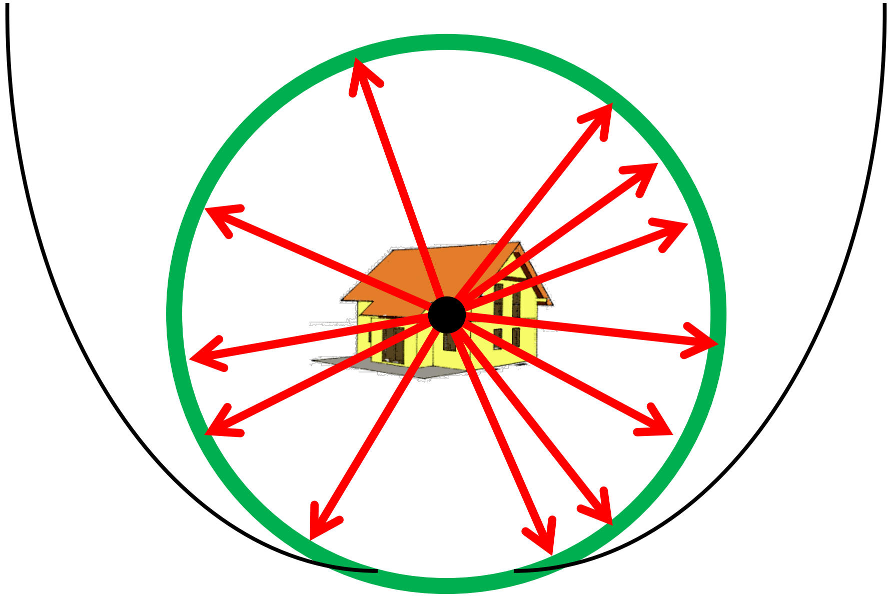


Environment Mapping Principle

= reflection mapping

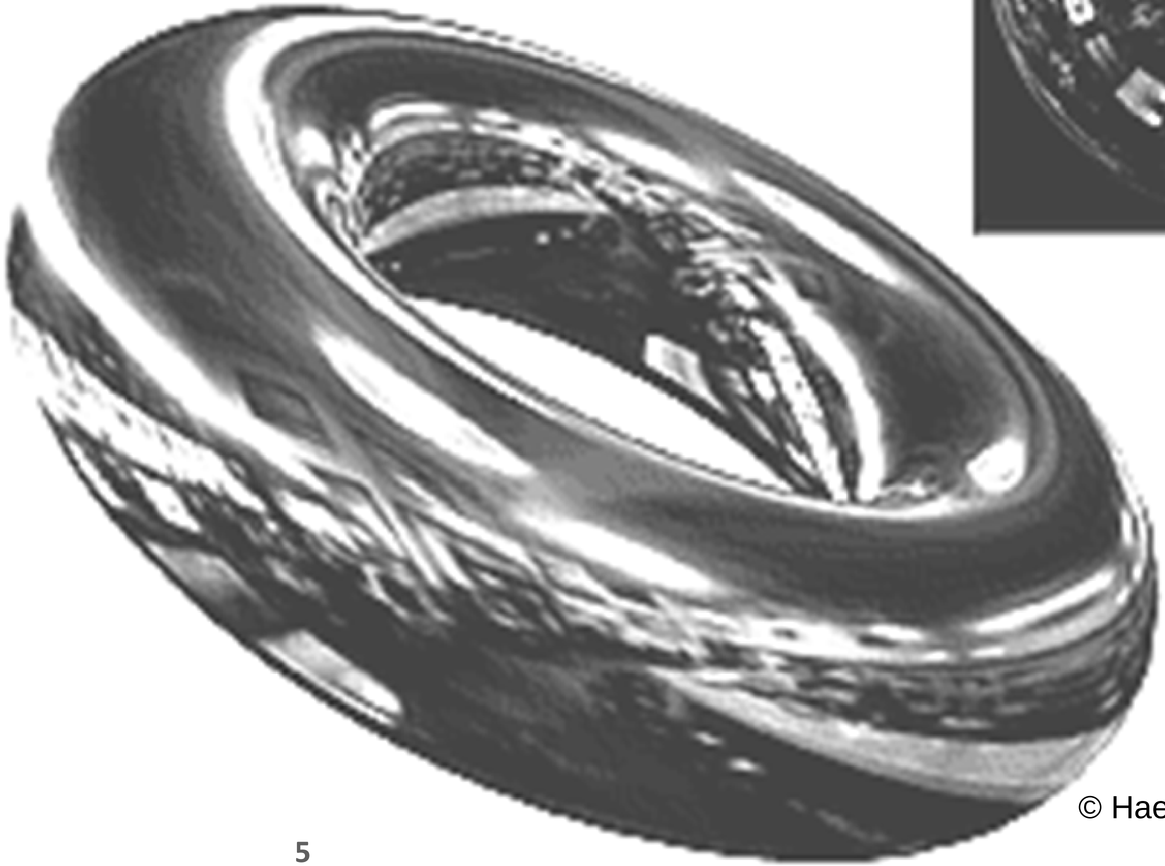
defined over surface of an enclosing universe (sphere, cube, cylinder)





Environment Mapping Examples

© Jim Blinn



© Haeberli/Segal

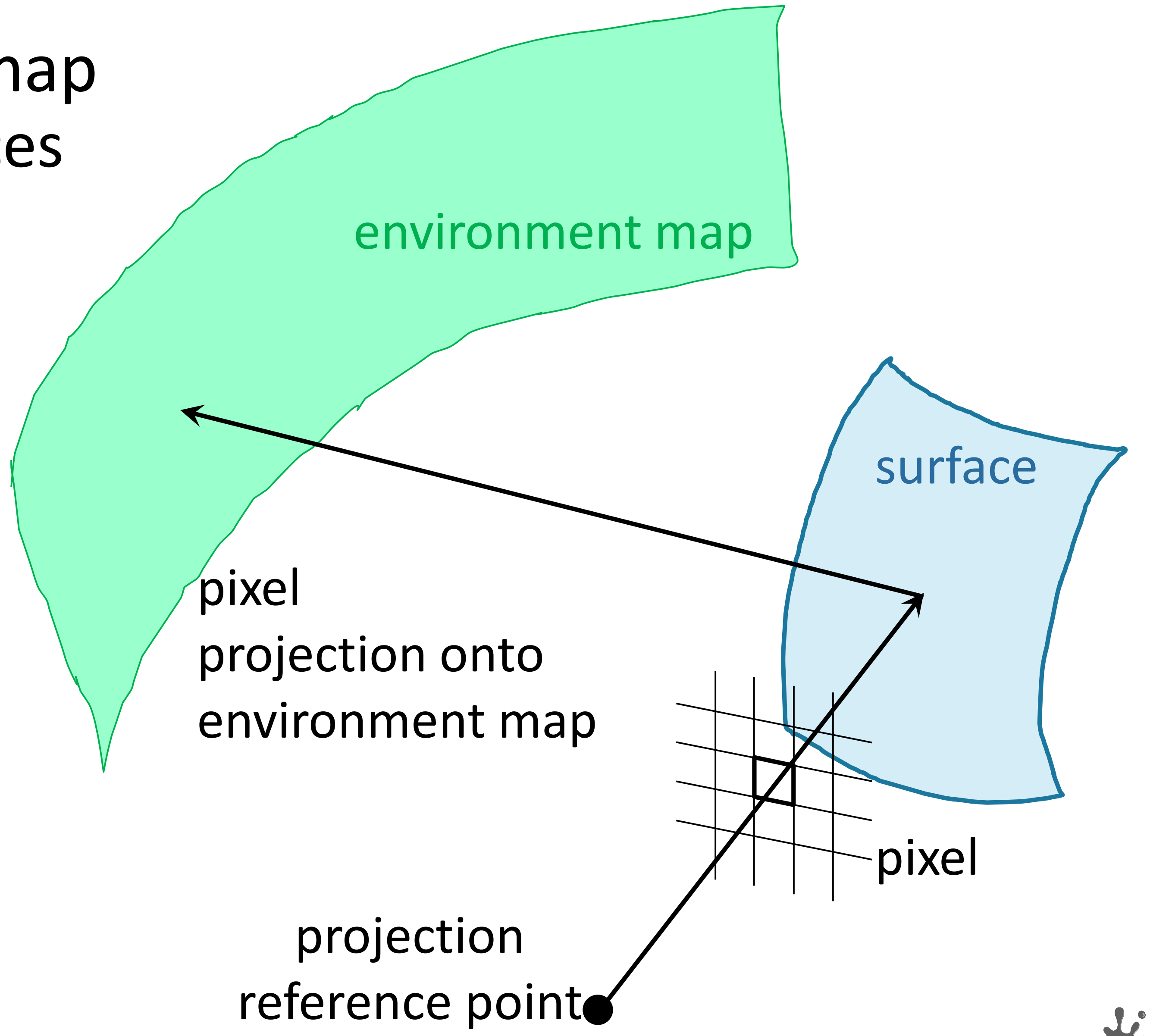
Environment Mapping Calculation

information in the environment map

- intensity values for light sources
- sky
- background objects

pixel:

- projected onto surface
- reflected onto environment map



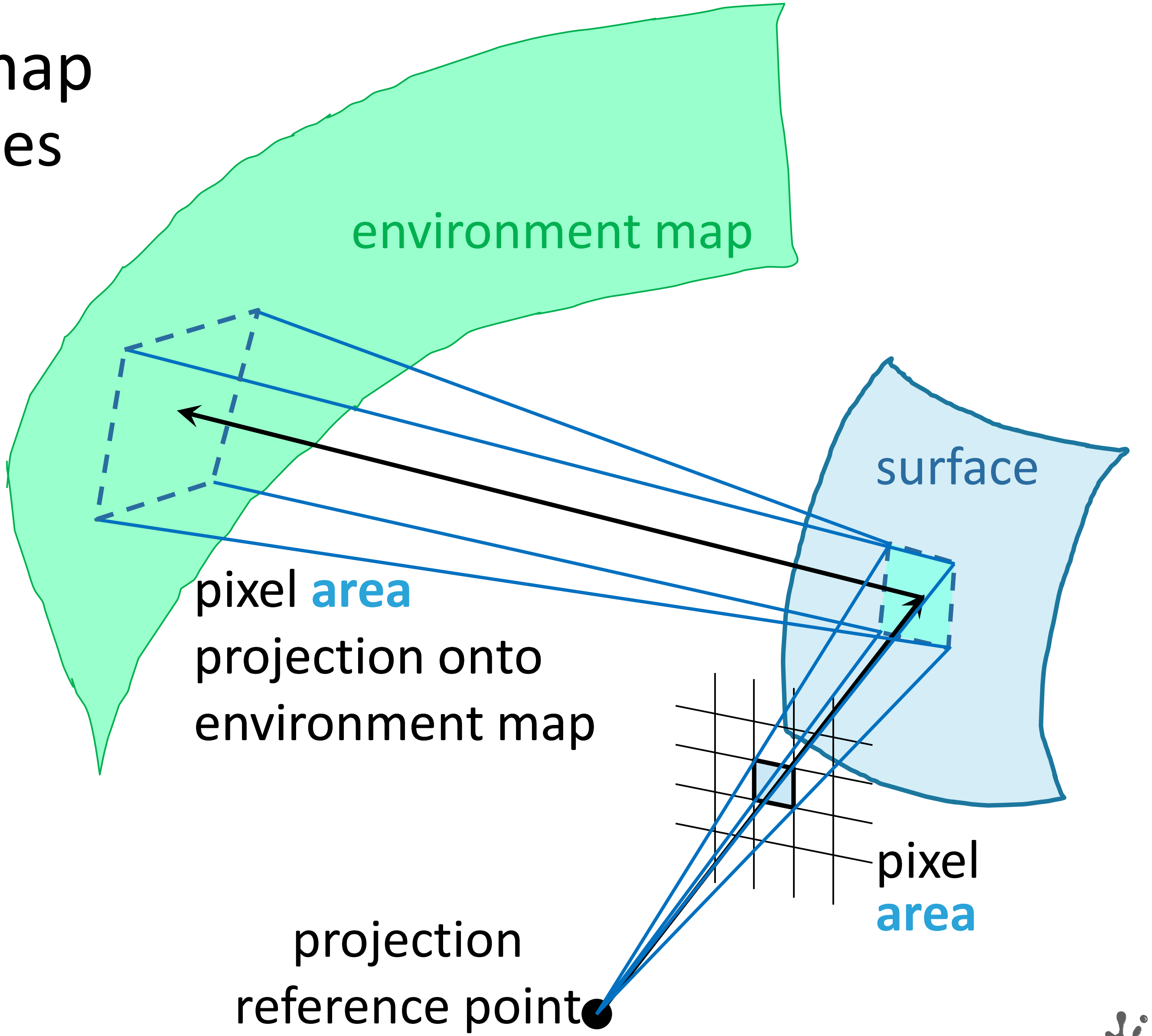
Environment Mapping Calculation

information in the environment map

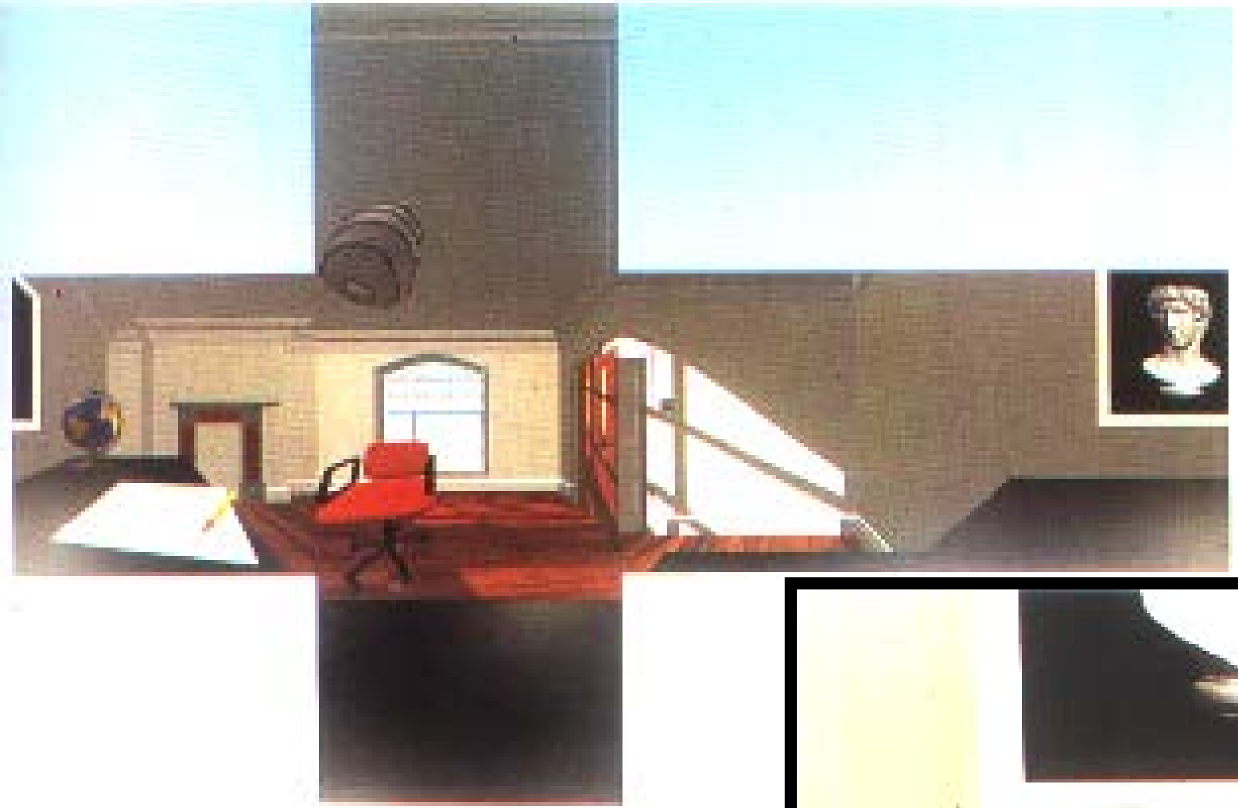
- intensity values for light sources
- sky
- background objects

pixel **area**:

- projected onto surface
- reflected onto environment map

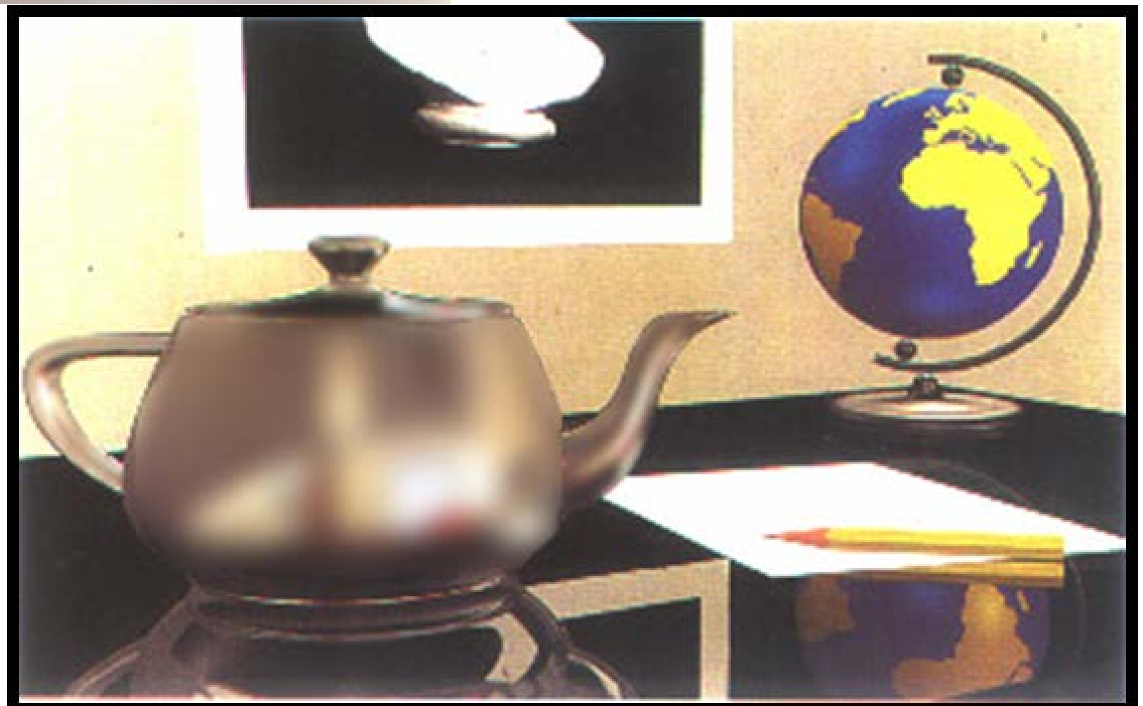


Environment Mapping Example

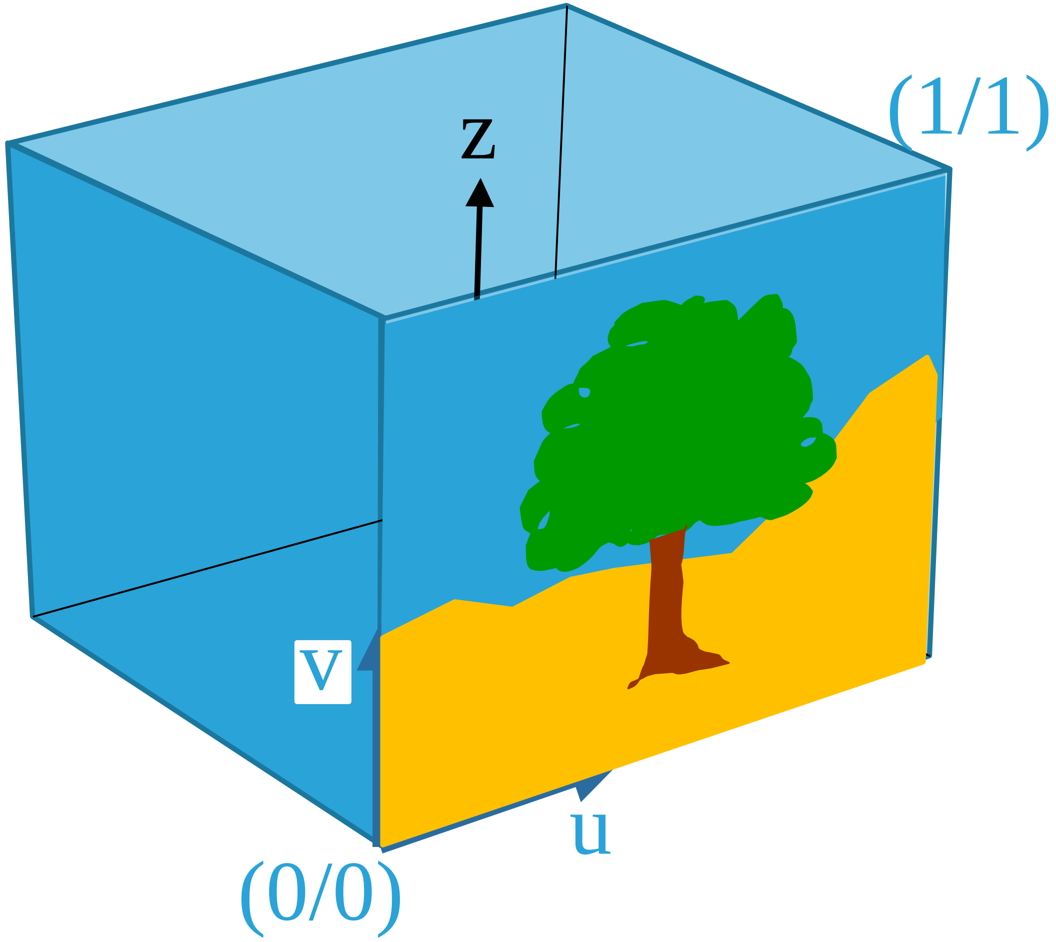


Environment Mapping Filtering

environment maps may be filtered for not so reflective surfaces



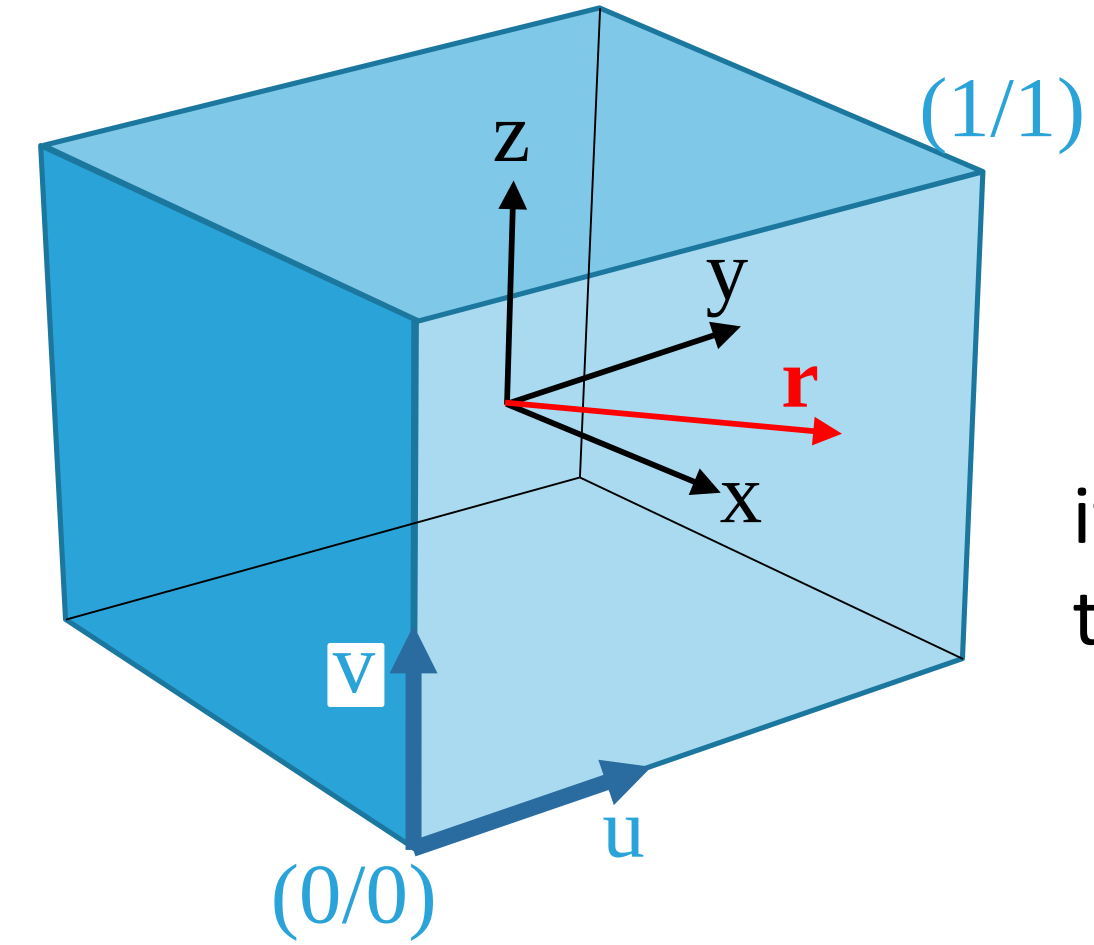
Environment Mapping with Cube-Map (1)



define textures on 6 sides of a cube, each parameterized in (u,v)

direction vector $\mathbf{r}$ starts at (0,0,0)

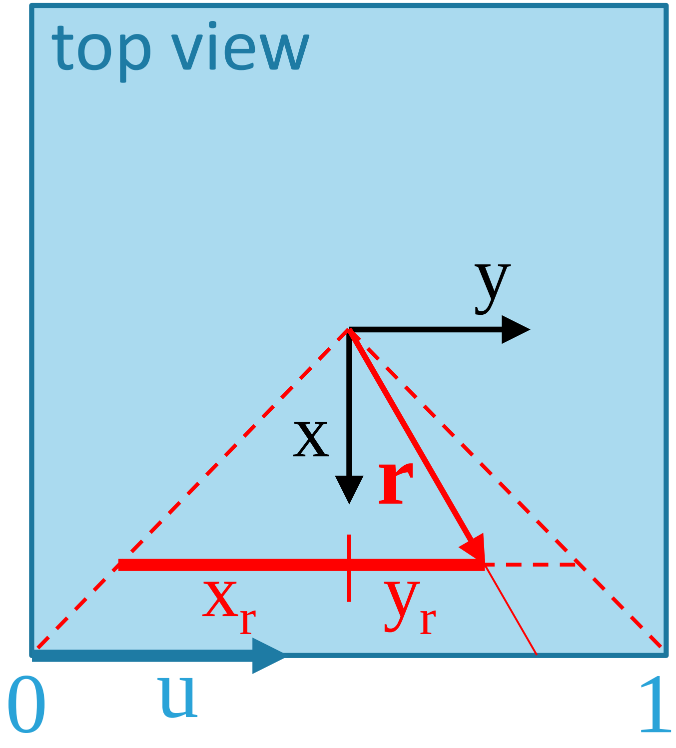
Environment Mapping with Cube-Map (2)



find (u,v) from the direction vector $\mathbf{r}(x_r,y_r,z_r)$:

if $x_r > |y_r|$ and $x_r > |z_r|$ then "front face"

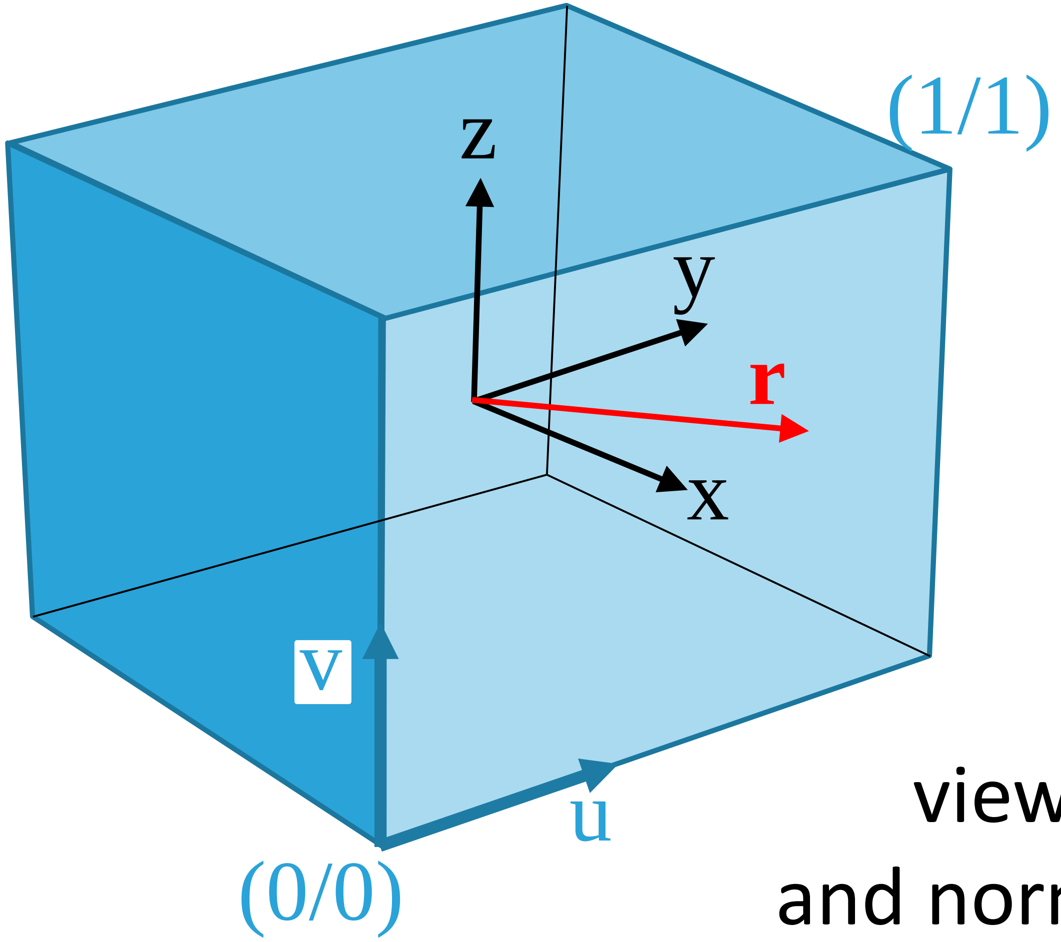
$$u = (y_r + x_r)/2x_r$$
$$v = (z_r + x_r)/2x_r$$



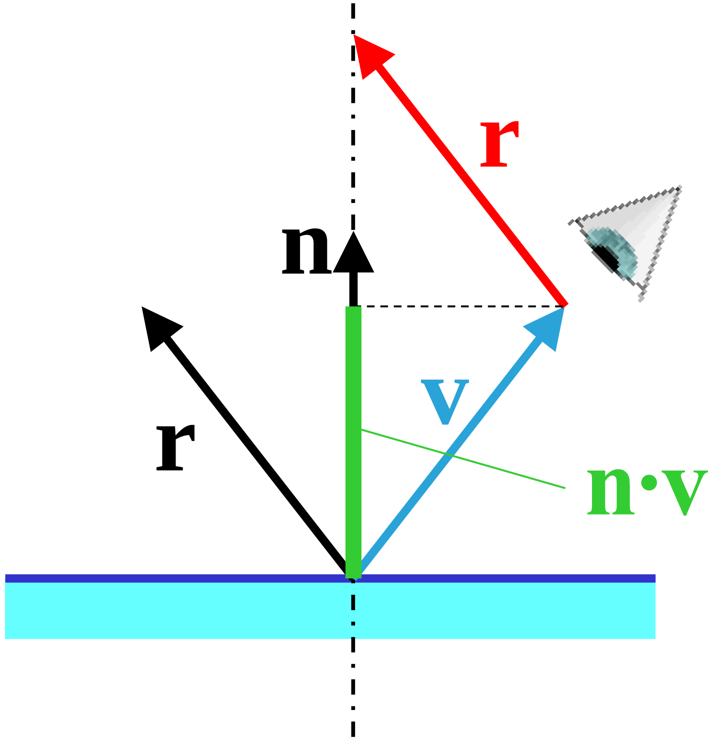
analogous formulas for the other 5 faces

$(-x>|y| \wedge -x>|z|, \quad y>|x| \wedge y>|z|, \quad -y>|x| \wedge -y>|z|, \quad z>|x| \wedge z>|y|, \quad -z>|x| \wedge -z>|y|)$

Environment Mapping with Cube-Map (3)



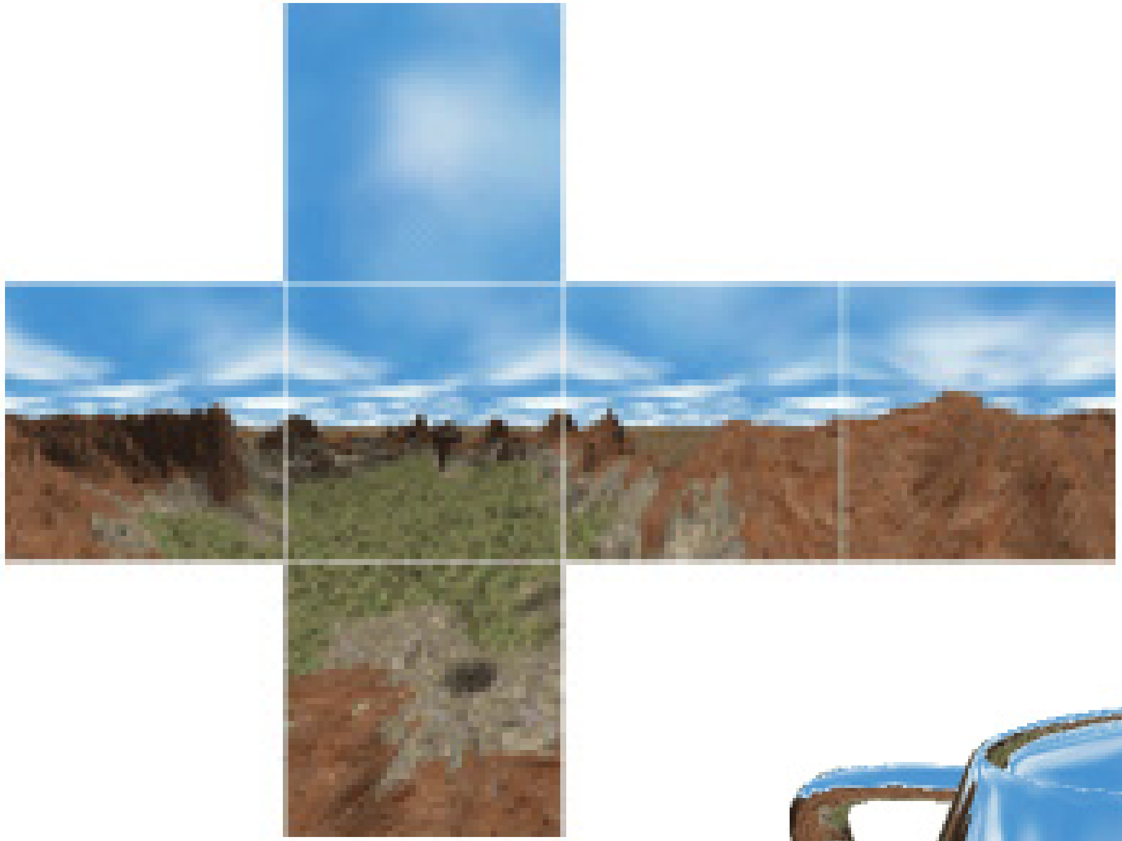
calculation of the direction vector **r**:



at a pixel:
viewing direction **v**
and normal vector **n** ⇒ $\mathbf{r} + \mathbf{v} = (2\mathbf{n}\cdot\mathbf{v})\mathbf{n}$

$$\boxed{\mathbf{r} = (2\mathbf{n}\cdot\mathbf{v})\mathbf{n} - \mathbf{v}}$$

Environment Mapping Example



Adding Surface Detail

most objects do not have smooth surfaces

- brick walls
- gravel roads
- carpets

→ surface texture required

Adding Surface Detail

modeling surface detail with polygons

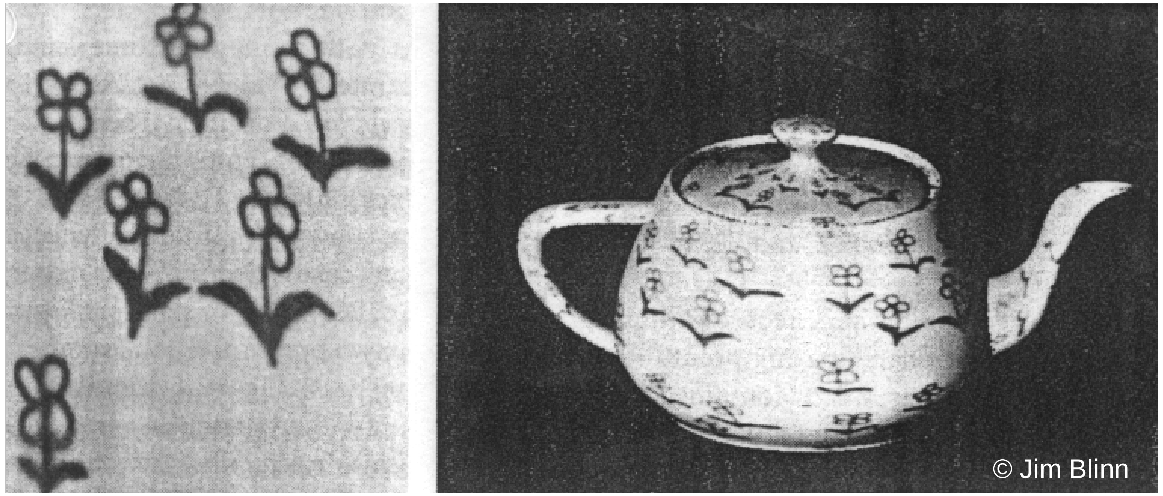
- small polygon facets (e.g., checkerboard squares)
- facets overlaid on surface polygon (parent)
- parent surface used for visibility calculations
- facets used for illumination calculations
- impractical for complicated surface structure

Texture Mapping: Principle

texture patterns are mapped onto surfaces

texture pattern can be:

- raster image
- or procedure
  (modifies surface intensities)



texture space → texture-surface transformation → **object space** → viewing & projection transformation → **image space**

Texture Mapping: Samples

Texture Mapping: Transformation

texture mapping options:

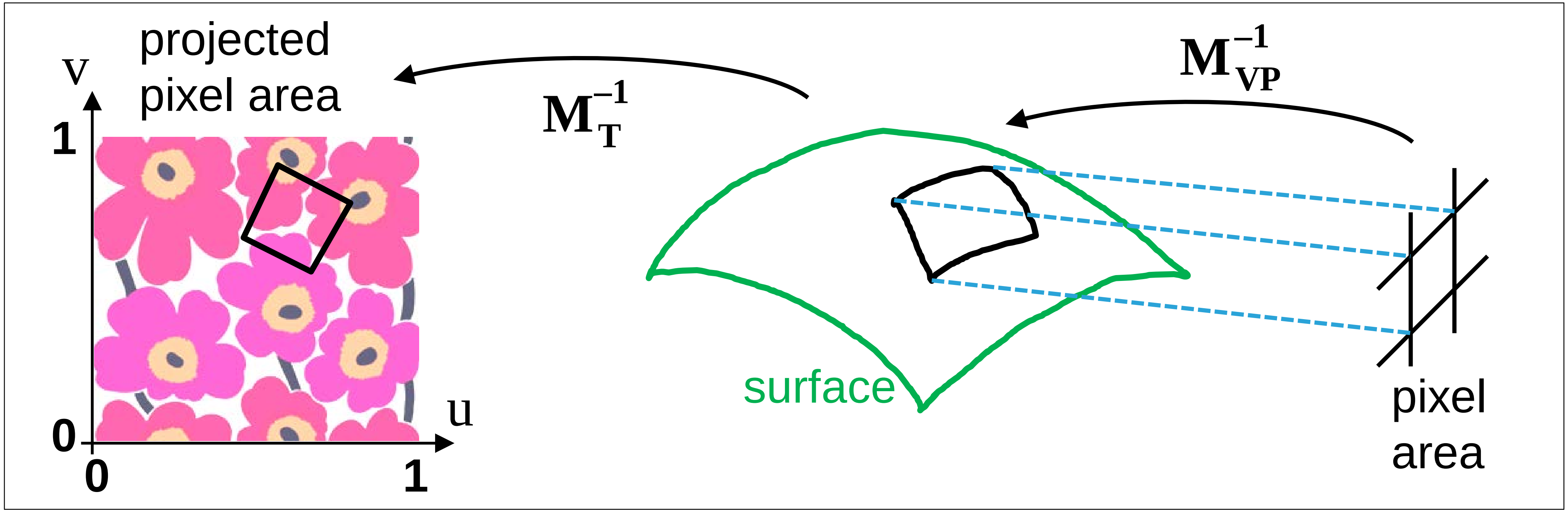
- texture scanning $(u,v) \to (x,y)$
- inverse scanning $(x,y) \to (u,v)$

texture space: (u,v) **array coordinates** $\xrightarrow[\text{transformation } M_T]{\text{texture-surface}}$ **object space:** (u^*,v^*) **surface parameters** $\xrightarrow[\text{transformation } M_{VP}]{\text{viewing \& projection}}$ **image space:** (x,y) **pixel coordinates**

texture-surface transformation:

$$u^* = u^*(u,v) = a_{u^*}u + b_{u^*}v + c_{u^*}$$
$$v^* = v^*(u,v) = a_{v^*}u + b_{v^*}v + c_{v^*}$$

projecting pixel areas to texture space = inverse scanning $(x,y)\rightarrow(u,v)$



- calculation of $\mathrm{M}_{\mathrm{VP}}^{-1}$ and $\mathrm{M}_{\mathrm{T}}^{-1}$
- anti-aliasing with filter operations

Texture-Mapping: Cylindrical Surface

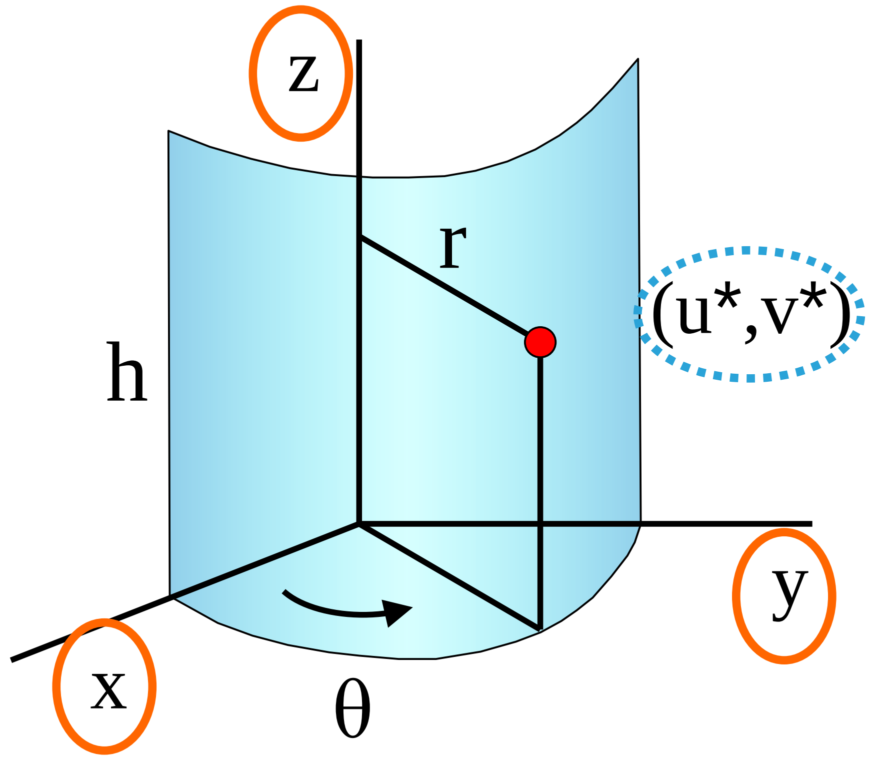
- M_{VP}

$x^2 + y^2 = r^2 \qquad x, y \geq 0 \qquad 0 \leq z \leq h$

$u^* = \theta$, with $0 \leq \theta \leq \pi/2$

$v^* = z$ with $0 \leq z \leq h$

$x = r \cdot \cos u^*$
$y = r \cdot \sin u^*$
$z = v^*$



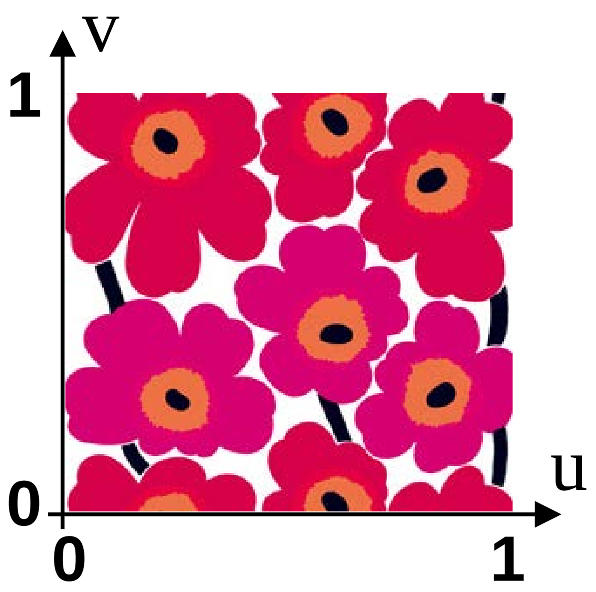
- M^{-1}_{VP}

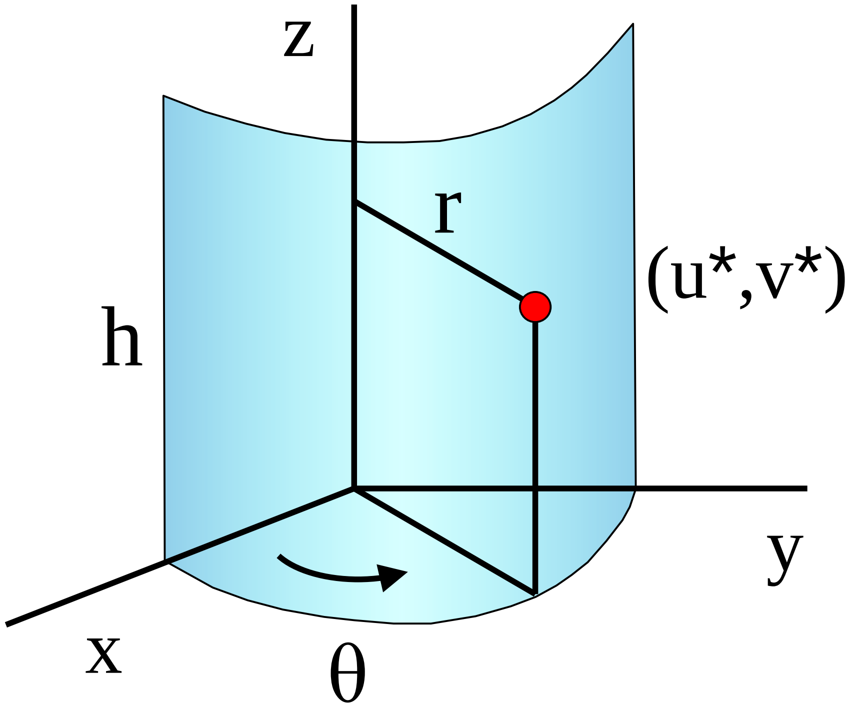
pixel $\rightarrow$ surface point (x,y,z)

$(x,y,z) \rightarrow (u^*,v^*)$: $\boxed{u^* = \cos^{-1}(x/r), \quad v^* = z}$

Texture-Mapping: Cylindrical Surface

- M_T $\quad u^* = u \cdot \pi/2, \quad v^* = v \cdot h$





- M^{-1}_T $\quad \boxed{u = 2u^*/\pi, \quad v = v^*/h}$

$$(u^* = \cos^{-1}(x/r), \quad v^* = z)$$

$$\boxed{u = 2\cos^{-1}(x/r)/\pi, \quad v = z/h}$$

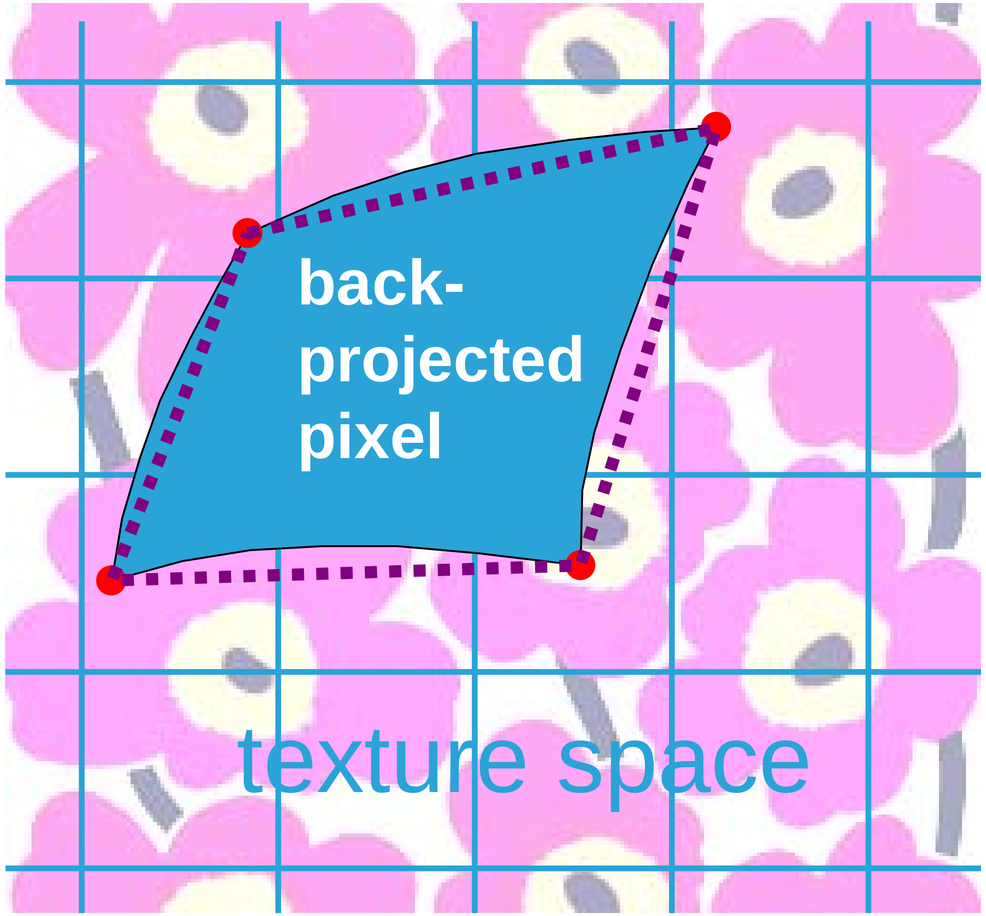
Texture Mapping: Anti-aliasing

anti-aliasing with filter operations:

project pixel area into texture space
and take average texture value

speed ups:

- mip-mapping
- summed-area table method

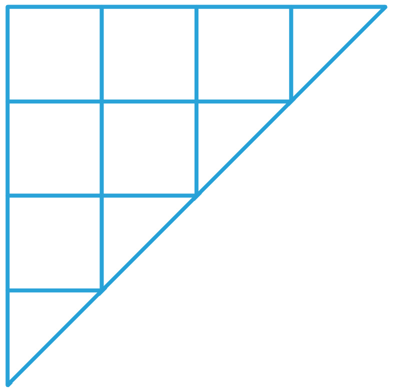


Perspective Correct Textures on Triangles

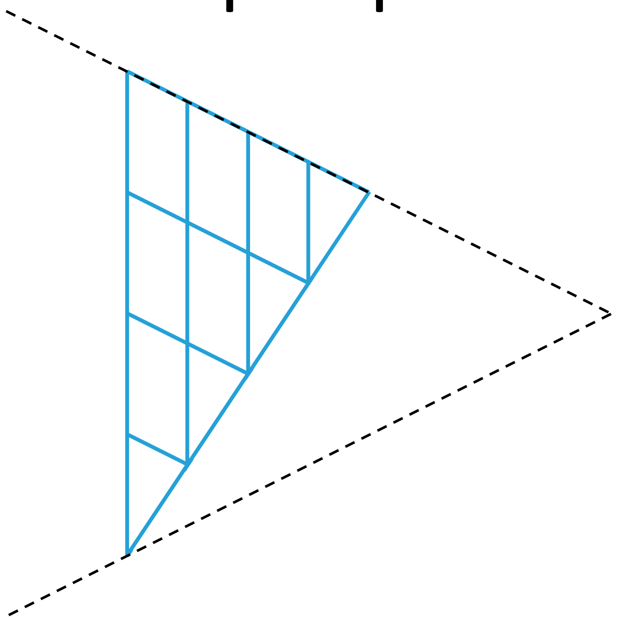
mapping a texture on a triangle with barycentric coordinates:

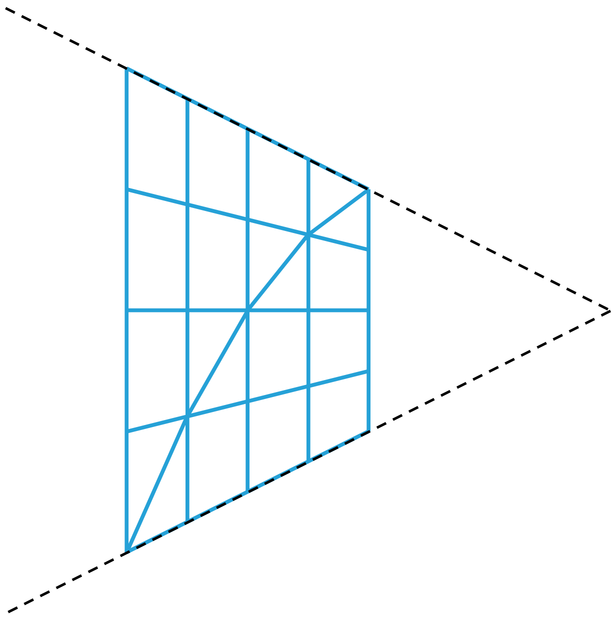
$$\mathbf{p}(\alpha,\beta,\gamma) = \alpha\cdot\mathbf{a} + \beta\cdot\mathbf{b} + \gamma\cdot\mathbf{c}$$

$$\text{color}(x,y) = \alpha\cdot t_0 + \beta\cdot t_1 + \gamma\cdot t_2$$

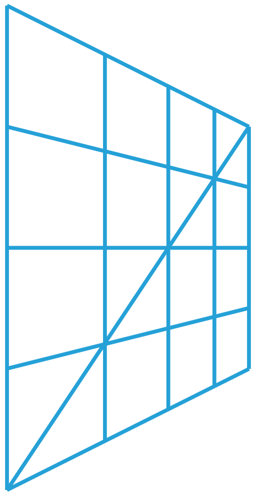


fails after perspective transform!





correct:



Perspective Correct Textures on Triangles

solution: keep homogeneous weights h_0 ,h_1 ,h_2 of **a, b, c**
and correct the barycentric values

$$d = h_1h_2 + h_2\beta(h_0 - h_1) + h_1\gamma(h_0 - h_2)$$

$$\beta_w = h_0h_2\beta/d \qquad \gamma_w = h_0h_1\gamma/d \qquad \alpha_w = 1 - \beta_w - \gamma_w$$

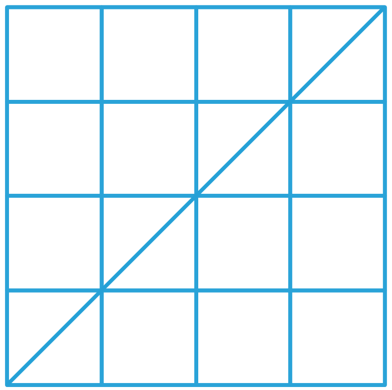
$$u = \alpha_w u_0 + \beta_w u_1 + \gamma_w u_2 \qquad v = \alpha_w v_0 + \beta_w v_1 + \gamma_w v_2$$

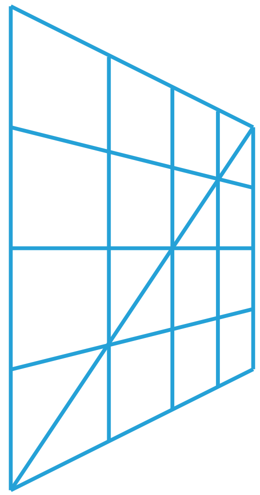
$$\text{color}(x,y) = t(u,v)$$

instead of

$$u=\alpha u_0+\beta u_1+\gamma u_2 \quad v=\alpha v_0+\beta v_1+\gamma v_2$$

$$\text{color}(x,y) = t(u,v)$$

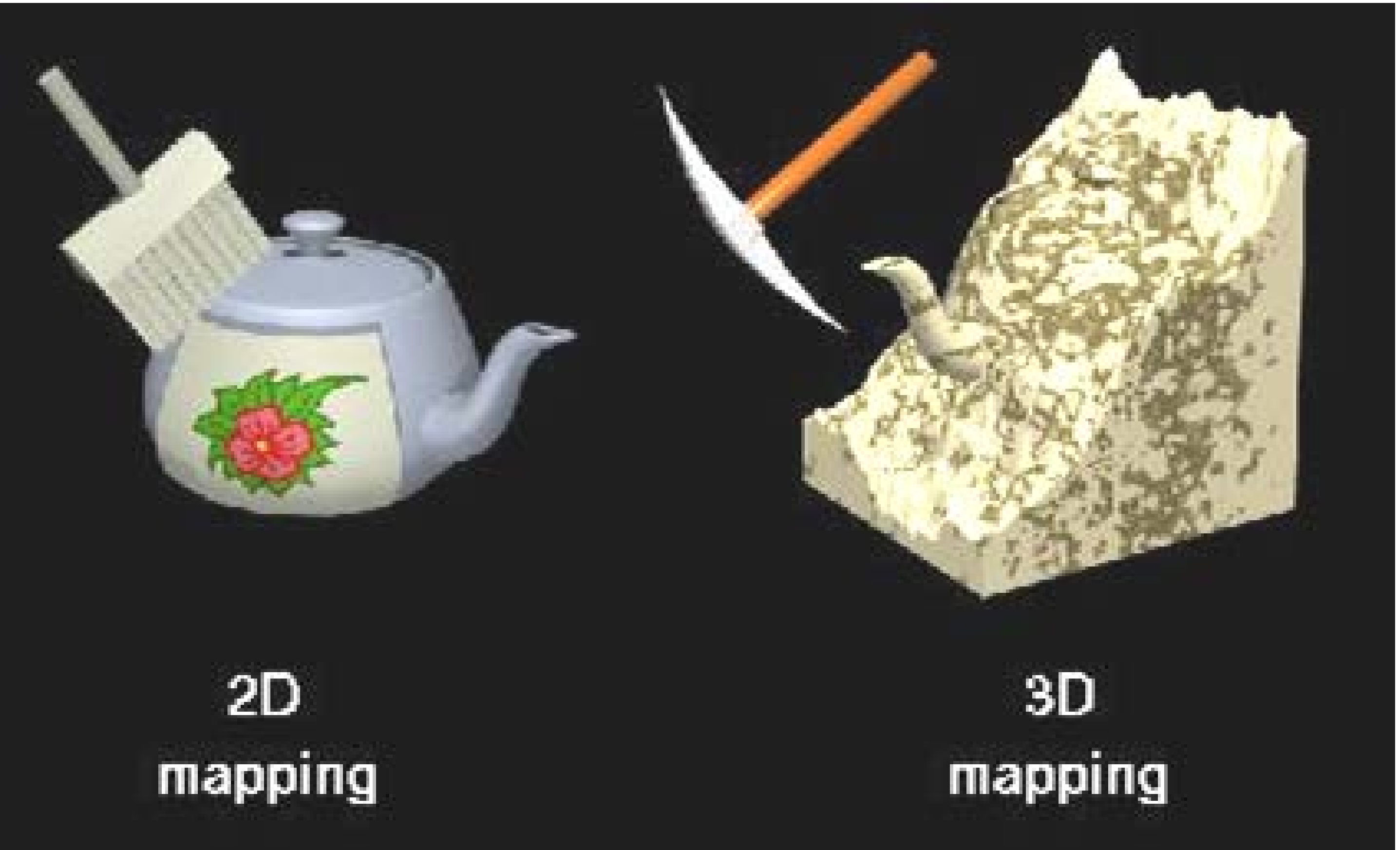




Solid Texturing

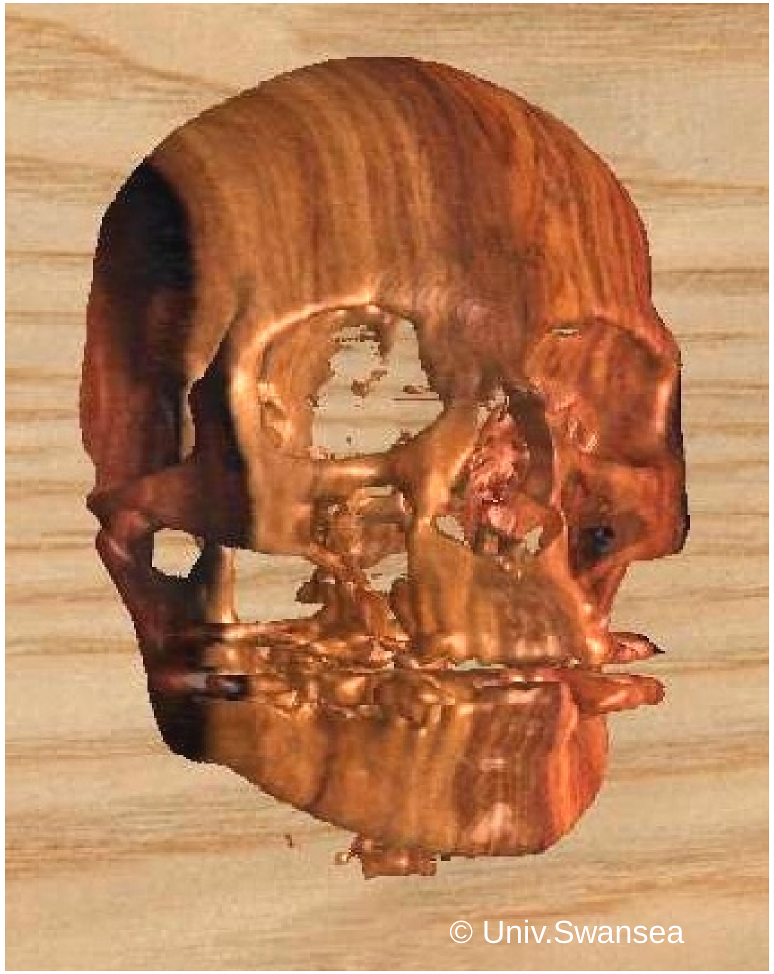
= texture defined in 3D

- every position in space has a color
- coherent textures across corners



Solid Texturing Examples

examples for application of 3D textures on a scull and a face



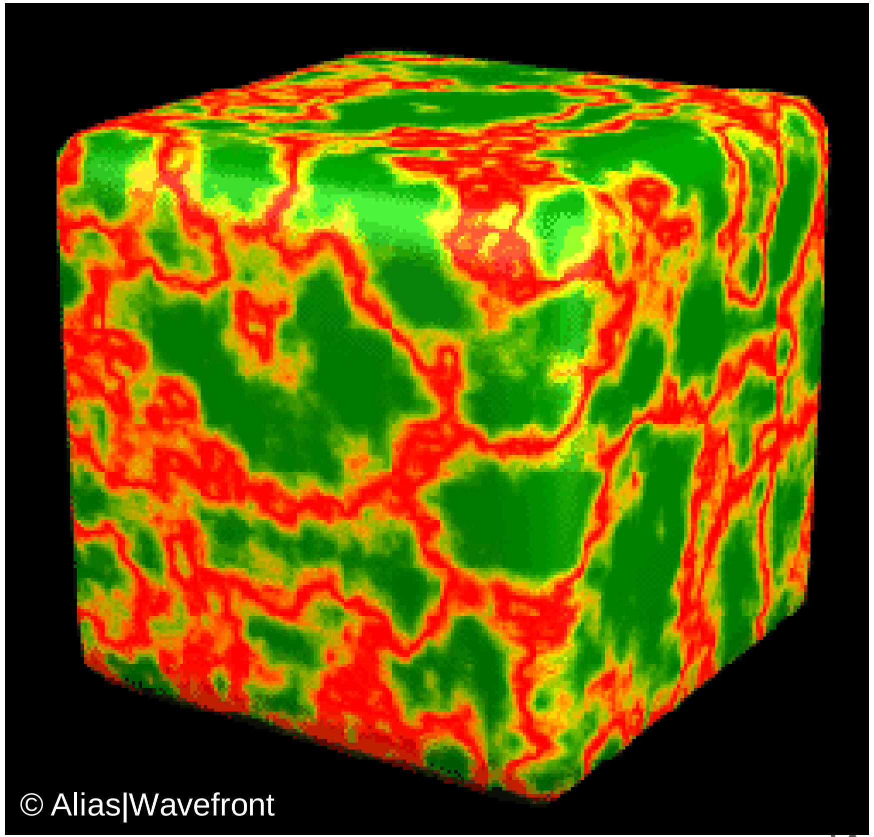
procedural texture definition

- texture-function $t(x,y,z)$ returns intensity
- avoid M_T

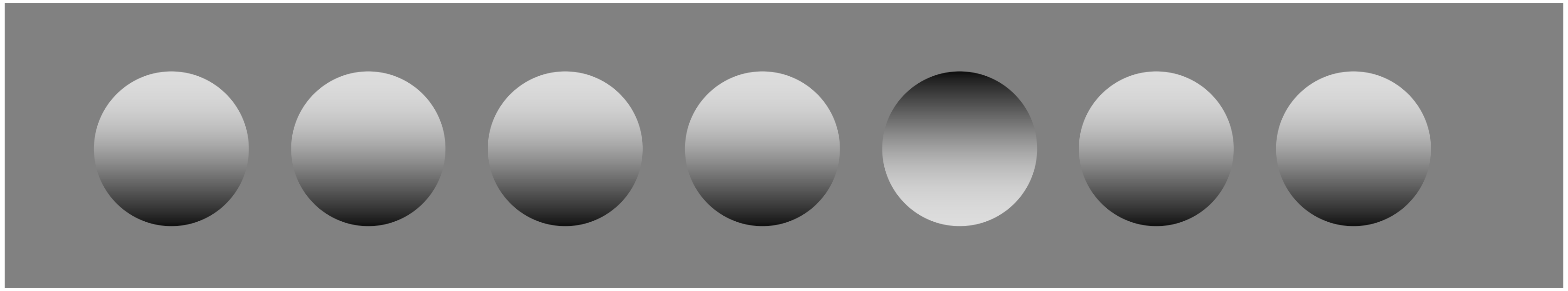
2D (surface texturing) or 3D (solid texturing)

stochastic variations (noise function)

examples: wood grains, marble, foam

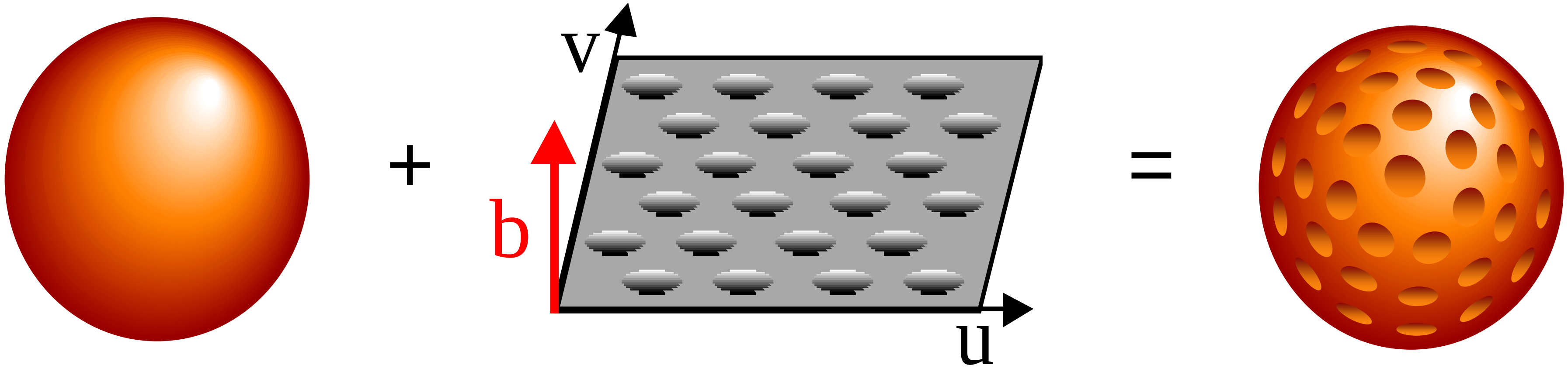


bumps are visible because of shading

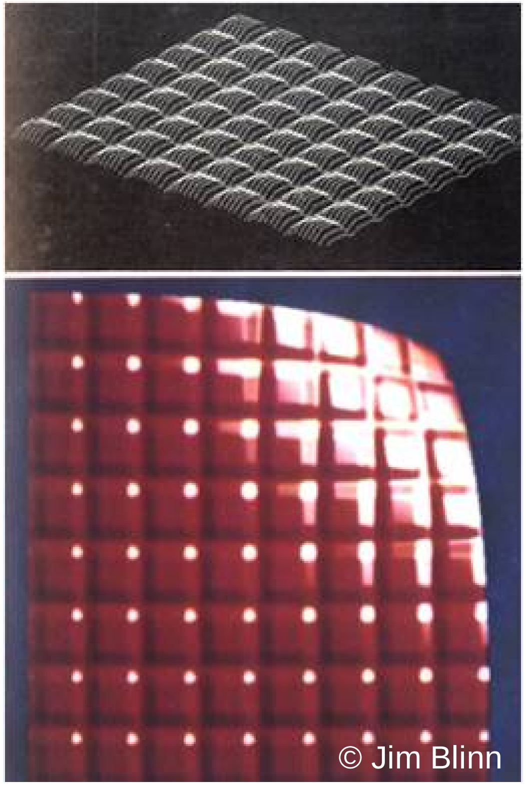


modeling of bumps is very costly.

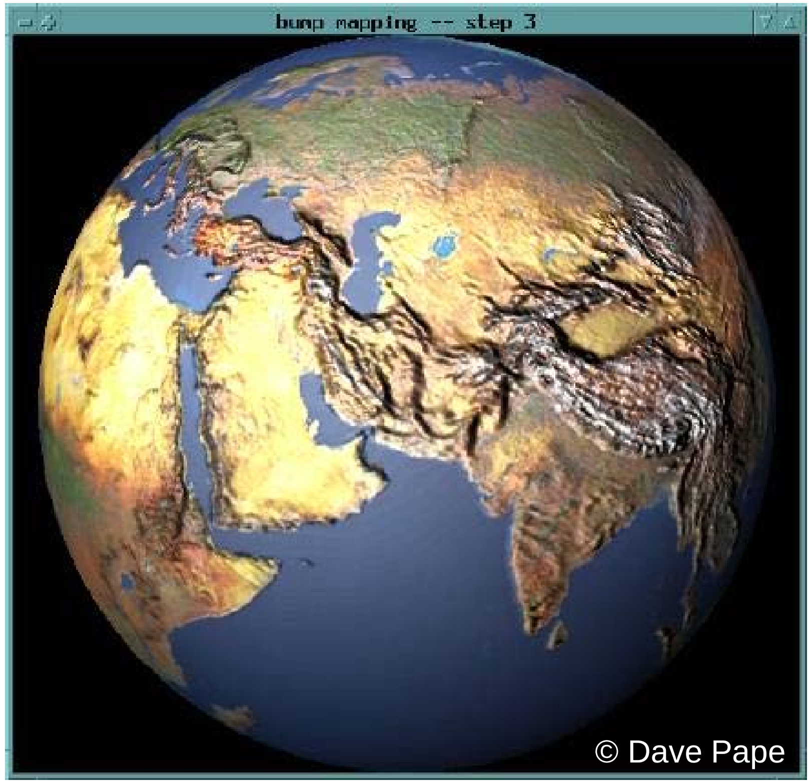
trick: insert a detail structure b:



Bump Mapping Examples

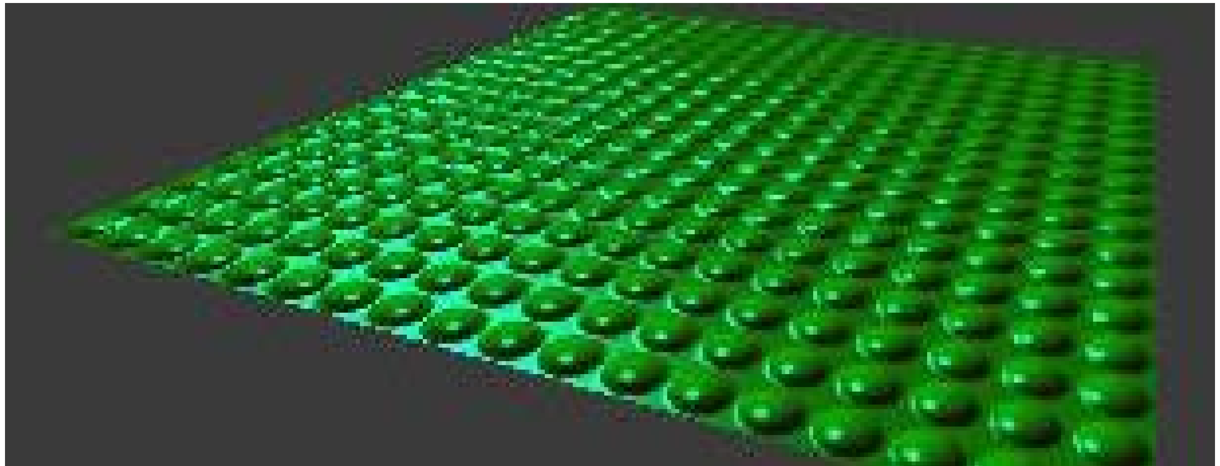


© Jim Blinn



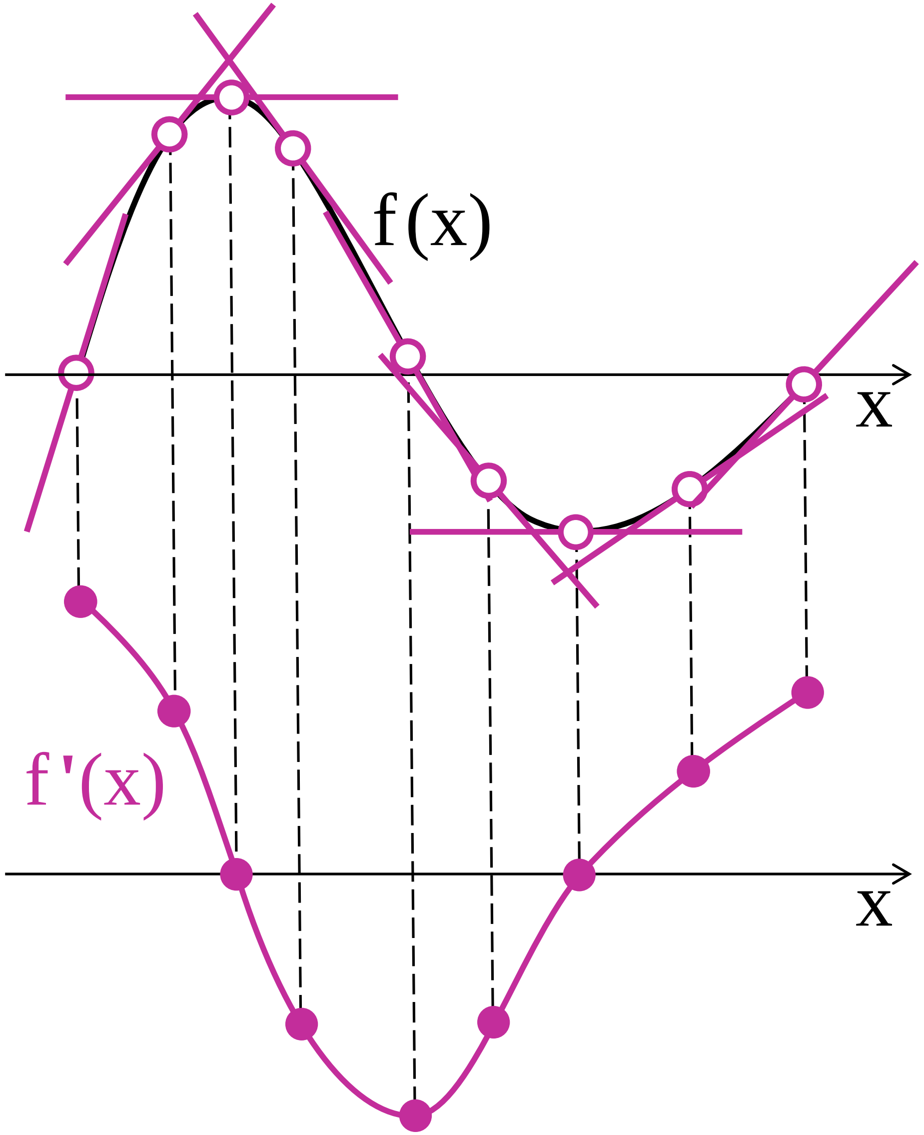
© Dave Pape

© Gamestar



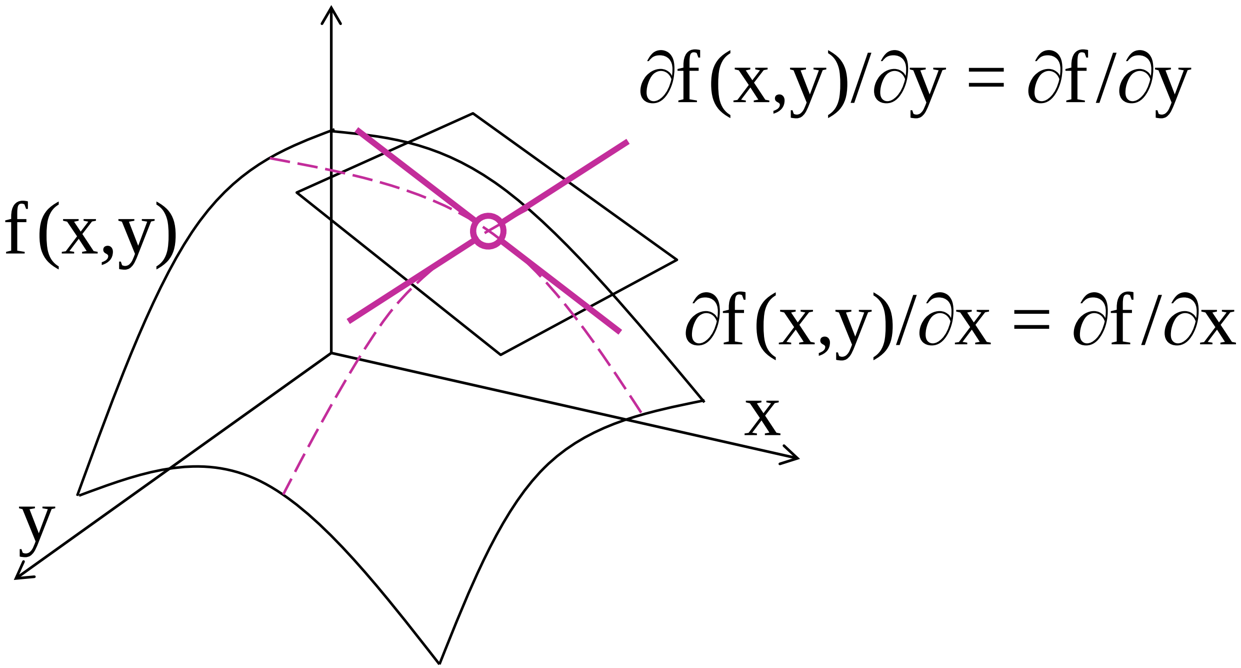
© Stanford

Reminder: Derivation of a Function



derivation of a function in one variable is the <u>*slope of the function*</u>

$$f'(x) = df(x)/dx = df/dx$$



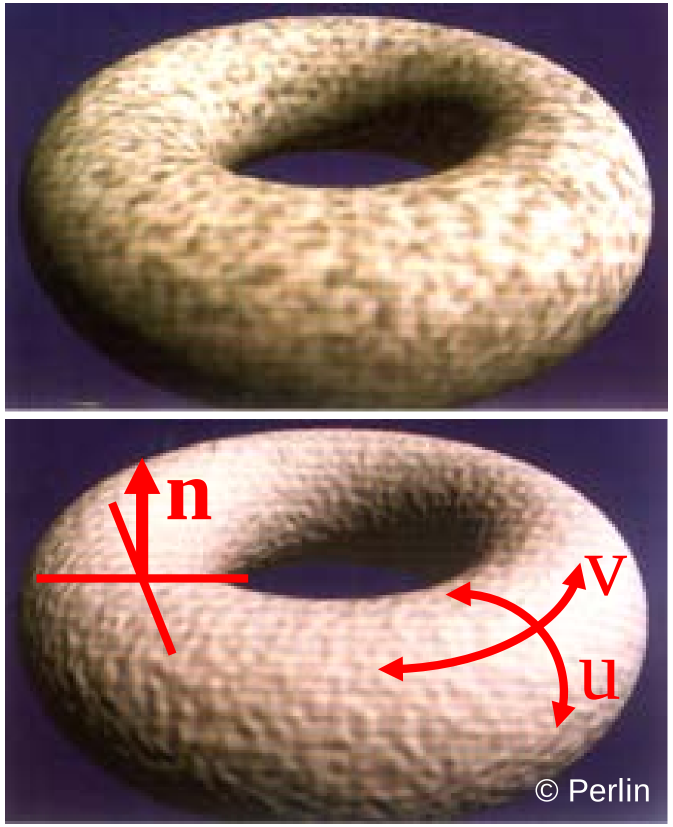
partial derivations of a function in two variables are the slopes of the functions when you keep one of the variables fixed, they are *slopes of tangents*

Bump Mapping Calculation

surface roughness simulated:

perturbation function varies surface normal locally

= **"bump map"** $b(u,v)$

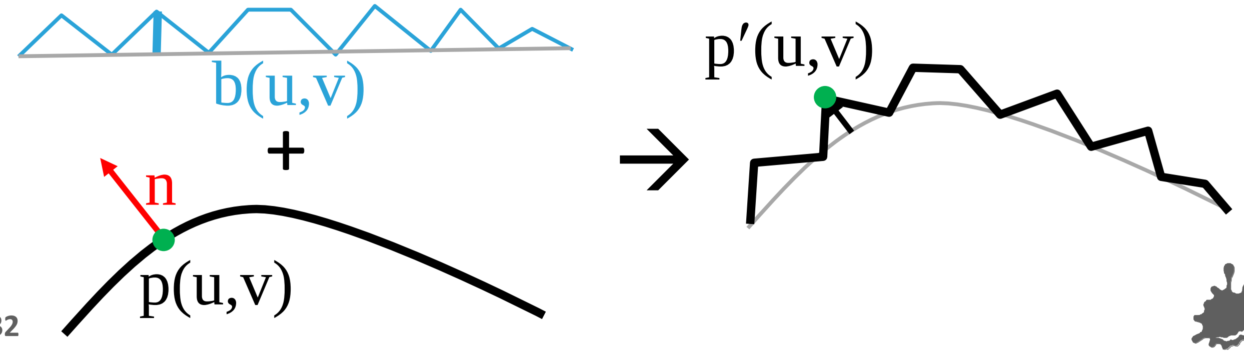


$\mathbf{p}(u,v)$ surface point

$\mathbf{n}^* = \mathbf{p_u} \times \mathbf{p_v}$ $\mathbf{n} = \mathbf{n}^* / |\mathbf{n}^*|$ surface normal

$\mathbf{p}'(u,v) = \mathbf{p}(u,v) + b(u,v) \cdot \mathbf{n}$

= modified surface point



Bump Mapping Calculation

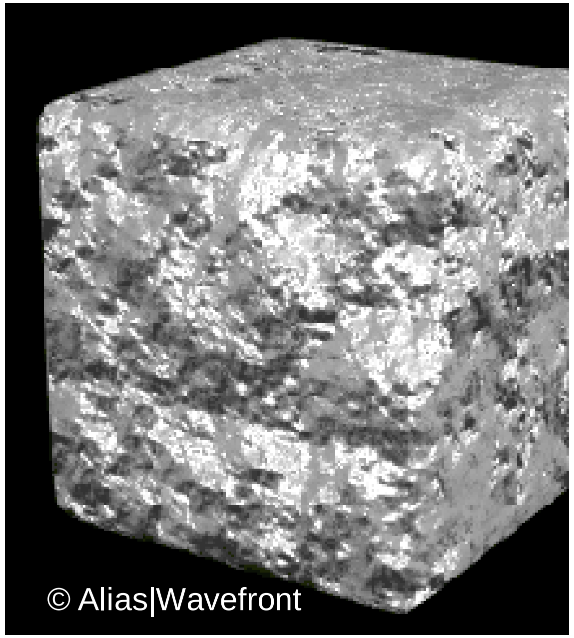
$$\mathbf{p}'(u,v) = \mathbf{p}(u,v) + b(u,v)\cdot\mathbf{n}$$

$$\mathbf{n}' = (\mathbf{p}_u' \times \mathbf{p}_v')$$

$$\mathbf{p}_u' = \partial(\mathbf{p}+b\mathbf{n})/\partial u = \mathbf{p}_u + b_u\mathbf{n} + b\mathbf{n}_u$$

$$\mathbf{p}_u' \approx \mathbf{p}_u + b_u\mathbf{n} \qquad \mathbf{p}_v' \approx \mathbf{p}_v + b_v\mathbf{n}$$

because
b is very small



$$\mathbf{n}' = (\mathbf{p}_u + b_u\mathbf{n}) \times (\mathbf{p}_v + b_v\mathbf{n})$$

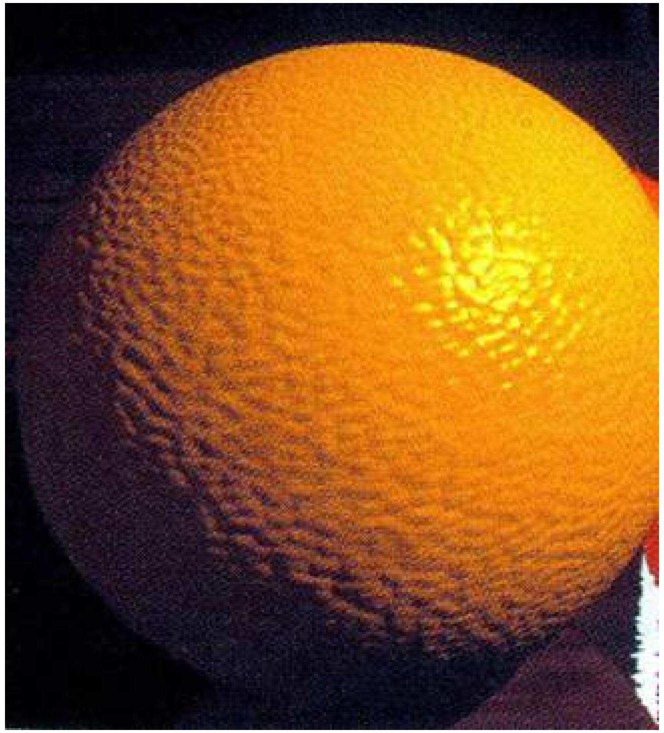
$$\mathbf{n}' = \mathbf{p}_u \times \mathbf{p}_v + b_v(\mathbf{p}_u \times \mathbf{n}) + b_u(\mathbf{n} \times \mathbf{p}_v) + b_u b_v(\mathbf{n} \times \mathbf{n}) \qquad \mathbf{n}\times\mathbf{n}=0$$

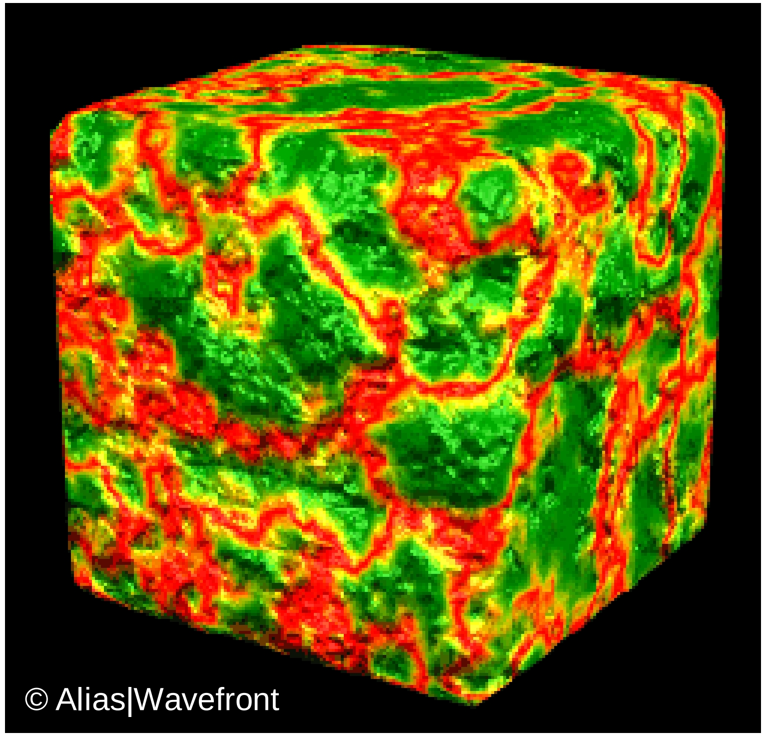
$$\boxed{\mathbf{n}' = \mathbf{n}^* + b_v(\mathbf{p}_u \times \mathbf{n}) + b_u(\mathbf{n} \times \mathbf{p}_v)}$$

Bump Mapping Representation

bump map $b(u,v)$ can be defined as raster image

b_u, b_v: approximated with finite differences





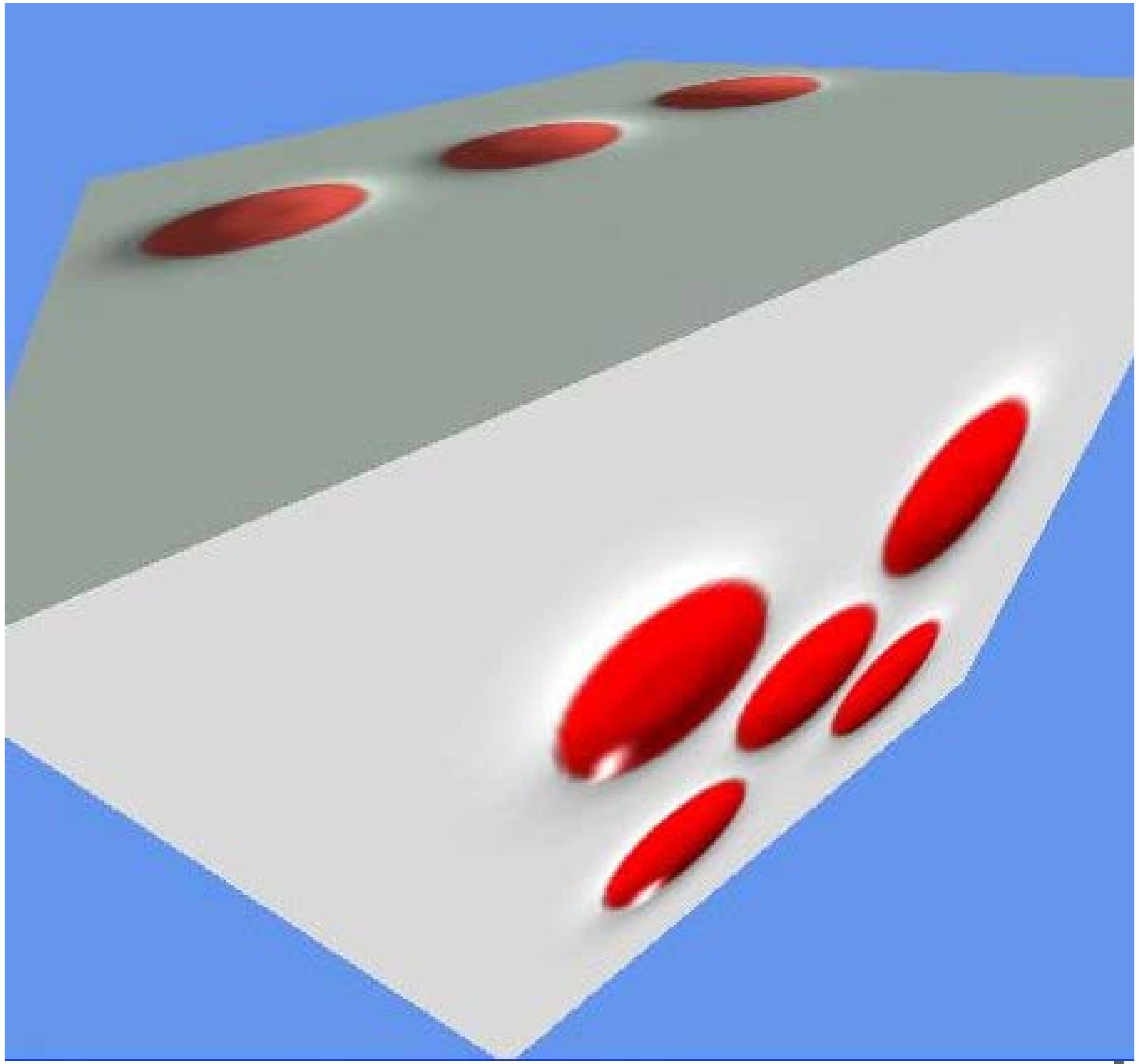
Bump Mapping Problems

sources of error:

- **distortions at grazing angles**
- wrong silhouette (geometry is not changed!)
- wrong shadows
- missing bump shadows
- light effects on back side

Bump Mapping: Grazing Angles

red buttons appear too flat,
although they are shaded in 3D



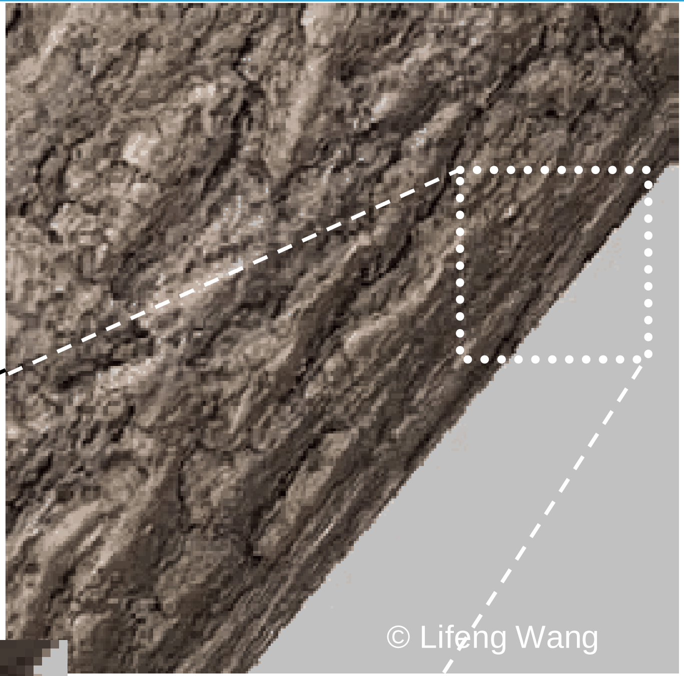
Bump Mapping Problems

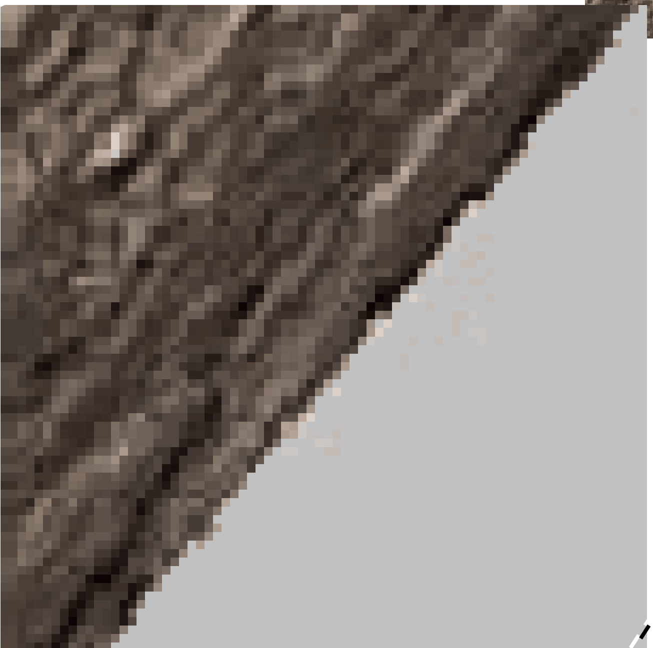
sources of error

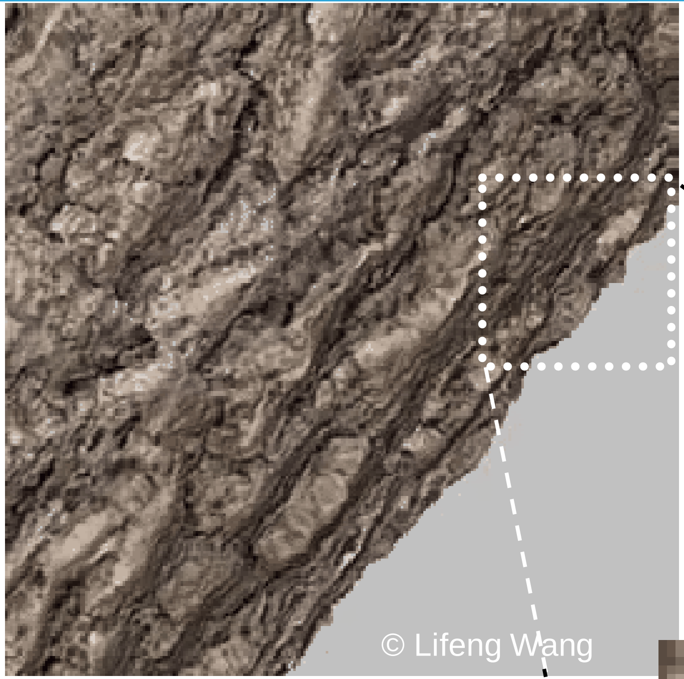
- distortions at grazing angles
- **wrong silhouette (geometry is not changed!)**
- wrong shadows
- missing bump shadows
- light effects on back side

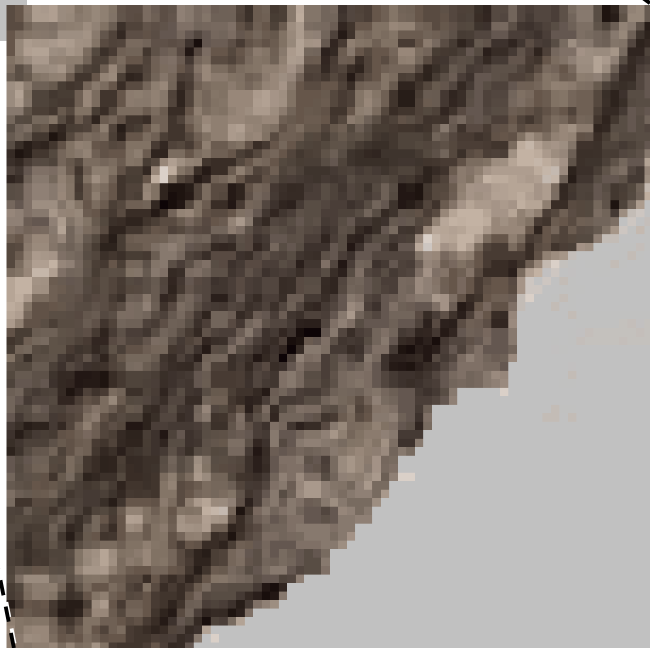
Bump Mapping: Wrong Silhouette

bump mapping









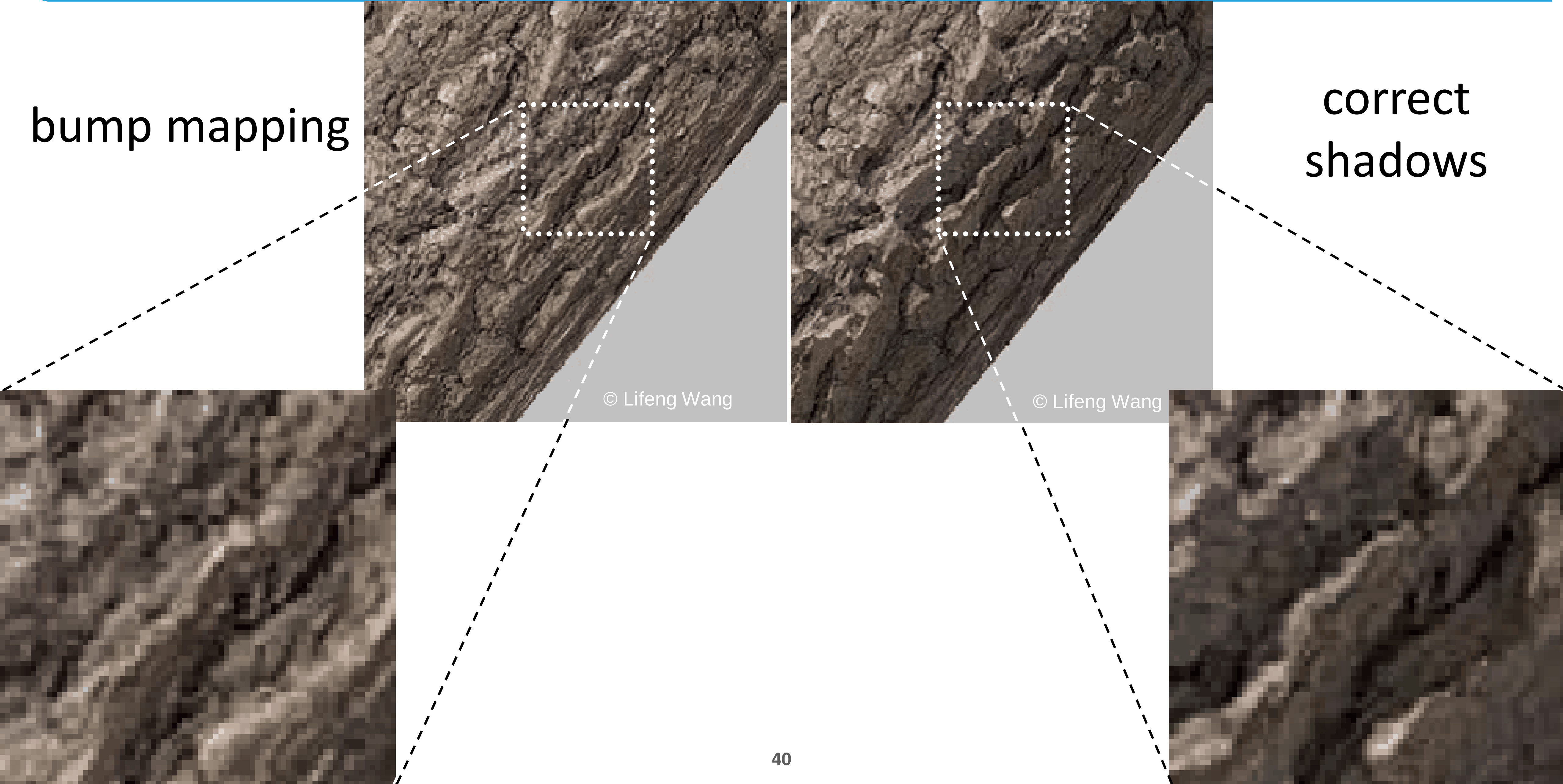
correct geometry

Bump Mapping Problems

sources of error

- distortions at grazing angles
- wrong silhouette (geometry is not changed!)
- **wrong shadows**
- **missing bump shadows**
- light effects on back side

Bump Mapping: Missing Bump Shadows

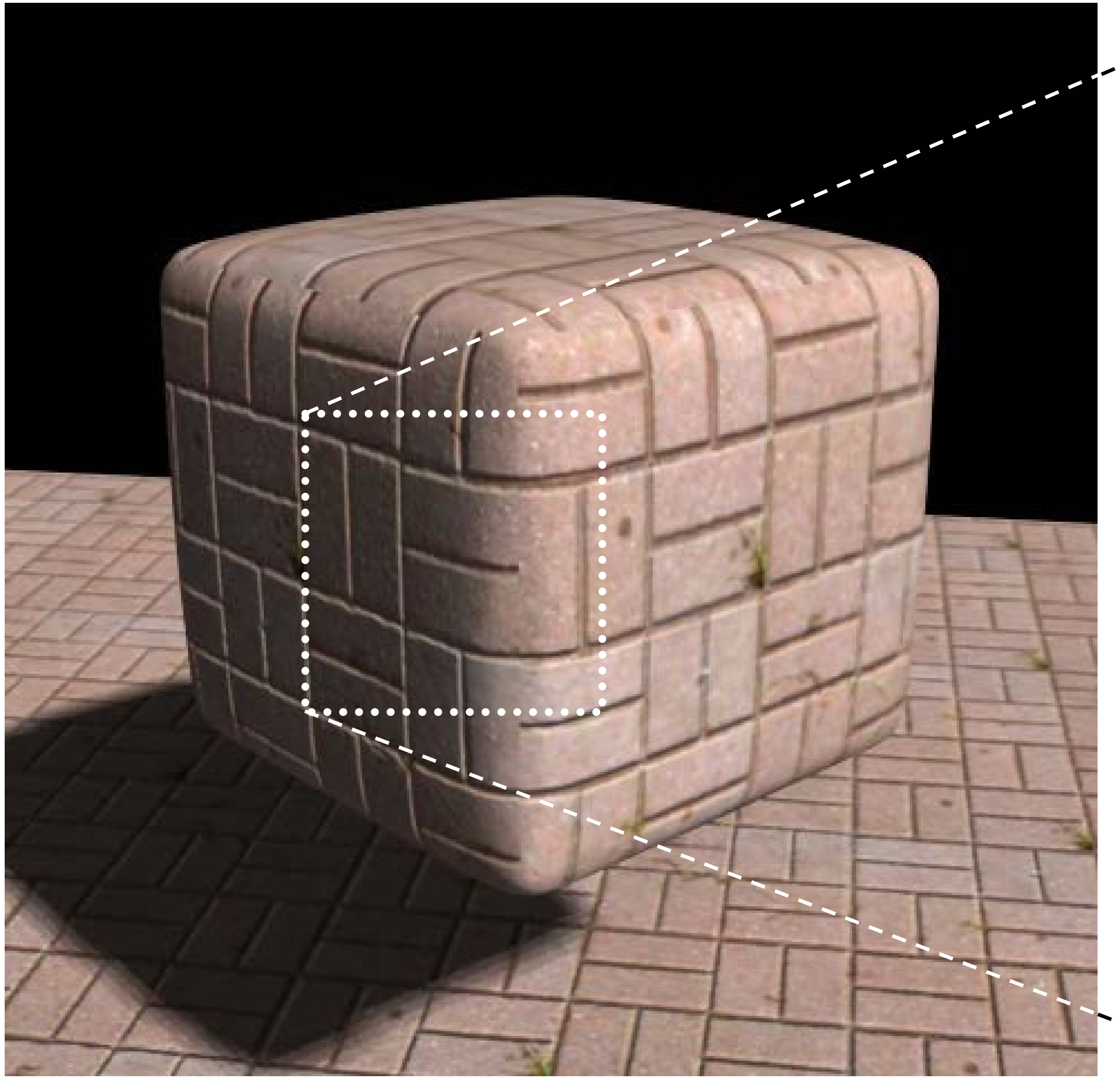


Bump Mapping Problems

sources of error

- distortions at grazing angles
- wrong silhouette (geometry is not changed!)
- wrong shadows
- missing bump shadows
- **light effects on back side**

Bump Mapping: Back Side Light Effects



Bump Mapping Problems

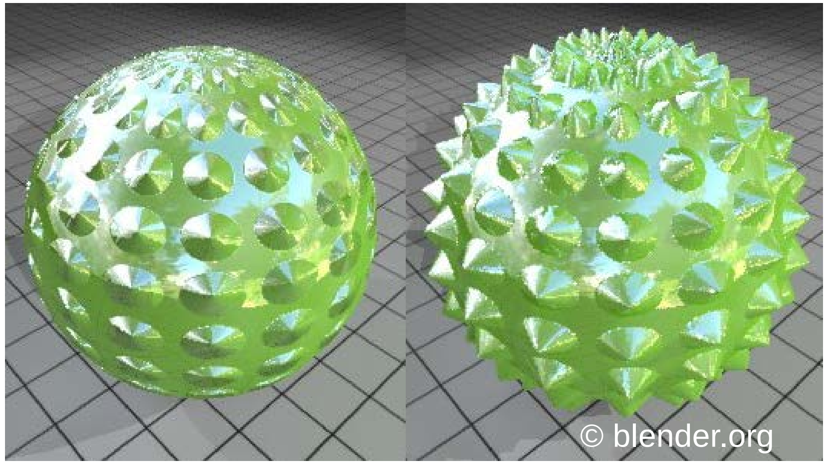
sources of error

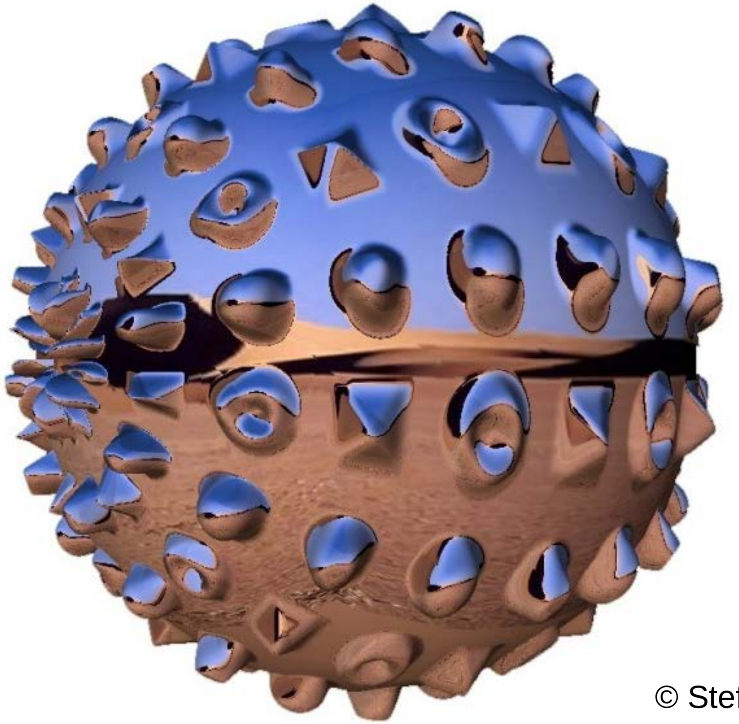
- distortions at grazing angles
- wrong silhouette (geometry is not changed!)
- wrong shadows
- missing bump shadows
- light effects on back side

→ there exist special algorithms to repair each error

Displacement Mapping

- “correct version of bump mapping”
- surface points are moved from their original position
- outline of object changes
- much harder to implement than bump mapping
  → rare in practice
- latest hardware partially supports it





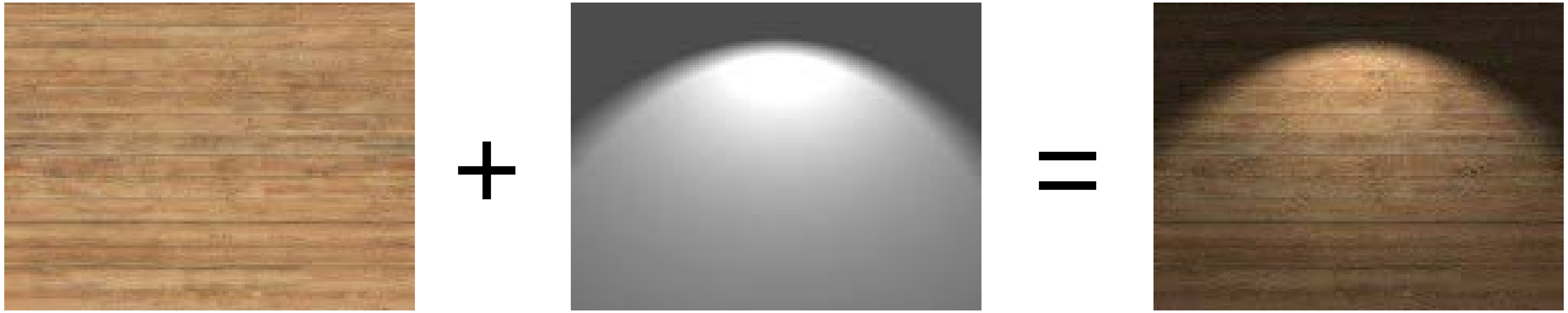
© Stefan Jeschke

Multitexturing: Combination of Mappings

= 2 or more textures applied to a surface

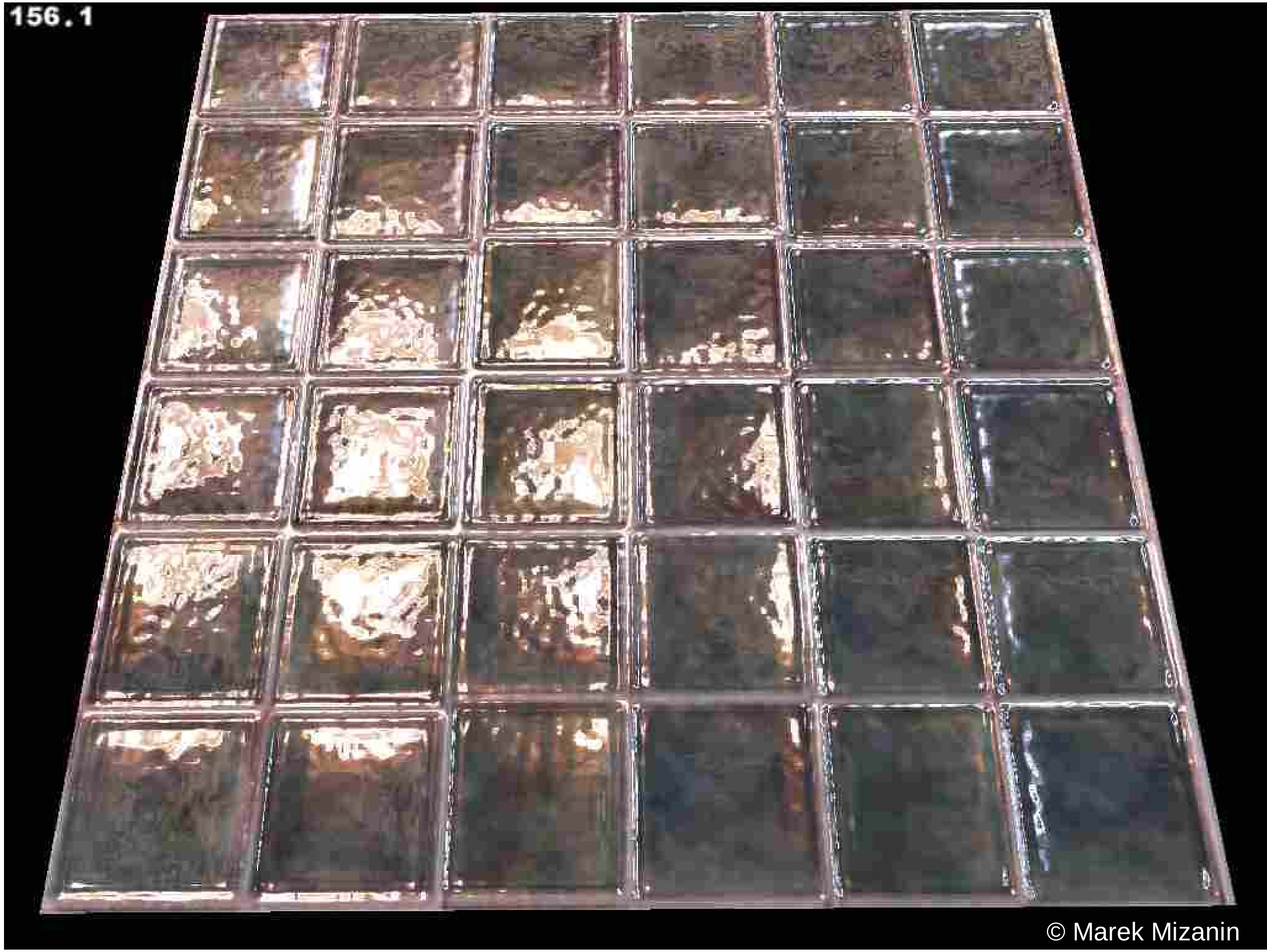
examples:

- texture & dirt
- texture & light map
- texture & bump map
- photo & annotations
- …



Multitexturing: Combination of Mappings

bump mapping
& environment mapping
& texture mapping



Einführung in Visual Computing

186.822

Textures

The End